21 世纪计算机系列规划教材

SQL Server 2008 数据库实用教程

姚一永　吕峻闽　主　编

靳紫辉　陈　婷　汤来锋　副主编

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内 容 简 介

本书以微软公司最新数据库产品 SQL Server 2008 为平台，在简明介绍传统数据库理论的基础上，详细阐述了 SQL Server 2008 系统的安装、使用和维护，强化实践教学和综合应用，并给出 VB 和 C#平台上完整的学生学籍管理系统开发过程，有利于读者对照学习，使读者能轻松自如地在 SQL Server 2008 平台上开发出一个完整的应用系统。

本书清晰描述了 SQL Server 2008 的主要功能及使用方法，突出重点和难点，同时注重对实际技能的培养。每章后都有习题并提供相关实验内容。

本书既可作为计算机及相关专业师生的教材，也可供数据库开发技术人员使用。

图书在版编目（CIP）数据

SQL Server 2008 数据库实用教程 / 姚一永，吕峻闽主编. —北京：电子工业出版社，2010.2
（21 世纪计算机系列规划教材）
ISBN 978-7-121-10279-0

Ⅰ. S… Ⅱ. ①姚…②吕… Ⅲ. 关系数据库－数据库管理系统，SQL Server 2008－高等学校－教材 Ⅳ. TP311.138

中国版本图书馆 CIP 数据核字（2010）第 014580 号

策划编辑：徐建军
责任编辑：徐　萍　文字编辑：徐　磊
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1 092　1/16　印张：16　字数：409.6 千字
印　　次：2010 年 2 月第 1 次印刷
印　　数：5 000 册　　定价：29.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

信息技术的飞速发展大大推动了社会的进步，也逐渐改变了人们的生活、工作和学习方式。数据库技术和网络技术是信息技术中的重要支柱。自 20 世纪 70 年代以来，数据库技术的发展已使得信息技术的应用从传统的计算方式转变为现代化的数据管理方式。在现代社会中，数据库技术的应用无处不在。当今各种热门的信息系统开发领域，如管理信息系统、企业资源计划系统、供应链管理系统、客户关系管理系统、电子商务系统、决策支持系统、智能信息系统等，都离不开数据库技术强有力的支持。据国际上有关机构统计，目前世界上存在一千多万个正在使用的数据库。Microsoft SQL Server 系统是一个典型的关系型数据库管理系统，它起步于 20 世纪 80 年代后期，是微软品牌中的一个重要产品。微软公司在 Microsoft SQL Server 产品方面投入了巨大的开发力量，持续不断地研发新技术以满足用户不断增长和变化的需求，从而使得该产品功能越来越强大，用户使用越来越方便，系统的可靠性越来越高，应用也越来越广泛。

本书主要基于对本科生实施大 IT 教育的理念，积极探索对非 IT 专业大学生进行 IT 教育的有效途径，结合近年来我们的教学和开发实践经验，以当前流行的 SQL Server 2008 数据库平台为实例，详略结合，突出基本，既汲取现有教学资料中合理的内容，又在对传统教学内容的介绍上有所创新。

全书共分为 12 章，内容涵盖了 SQL Server 2008 系统应用的方方面面，不仅包括数据库经典理论的介绍，还包括 SQL Server 2008 数据库产品的详细安装方法，数据库和数据表的创建、修改和查询，T-SQL 语言的使用方法，存储过程，事务处理，数据完整性，数据备份和安全性管理等高级应用。对于各个知识点的讲解，都配有大量可实际运行的实例，可供读者边学习边实践，以方便读者快速、全面地掌握 SQL Server 的使用方法和技巧。

本书由姚一永、吕峻闽主编，靳紫辉、陈婷、汤来锋任副主编并负责编写相应各章节，郭进负责应用部分的程序设计。参加本书编写的还有黄纯国、杨大友、陈斌、陈小宁、王静、王玉晶、张英、宁涛等。同时西南财经大学天府学院信息技术教研中心和现代技术中心的各位老师为本书提供了许多帮助，在此，编者对以上人员致以最诚挚的谢意！在编写本书的过程中参考了相关的图书和资料，在此也对这些资料的相关作者深表感谢。

为了方便教学，可登录 www.hxedu.com.cn 免费下载与本书配套的教学资源。

由于编者水平有限，加之时间仓促，书中难免有不妥之处，敬请读者批评指正，以便在今后的修订中不断改进。

编　者

本书编委会成员

（按拼音排序）

陈　斌　　陈　康　　陈　婷　　陈小宁　　段华薇

冯　亮　　韩延明　　何　亮　　黄纯国　　靳紫辉

李长松　　李玲鞠　　梁浴文　　吕峻闽　　宁　涛

邱　伟　　汤来锋　　王　静　　王　强　　王玉晶

吴希敏　　肖　忠　　谢志龙　　夏钰红　　杨大友

姚一永　　袁　勋　　张　笑　　张　英　　赵　华

目　　录

第 1 章　数据库基本概念和 SQL Server 2008 环境

本章先主要介绍数据库的基本概念、数据管理的发展过程、数据模型的基本知识、数据库系统的组成部分，接着介绍 SQL Server 2008 的入门知识，包括 SQL Server 2008 的特点、安装方法、文件系统，以及它的管理工具和配置方法等。

通过对本章的学习，可以了解为什么要使用数据库，以及数据技术的重要性，也可为以后章节的学习打下基础。

- 理解关系数据库的组成和基本概念
- 理解实体与关系模型的概念
- 掌握安装 SQL Server 2008 的方法
- 掌握配置 SQL Server 2008 的方法
- 了解 SQL Server 2008 的重要管理工具

1.1　数据库系统的基本概念

当人们收集大量数据，并抽取出某个应用所需要的数据时，应将其保存起来以供进一步加工处理，保存数据的地方则称为数据库（Database，DB），即存放数据的仓库，只是这个仓库是建立在计算机的存储设备上的。在计算机中大量的数据通常不会被随机存放，而是按一定的格式有组织地存放，所以数据库的定义是长期储存在计算机内、有组织的、可共享的大量数据的集合。数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度、较高的数据独立性和易扩展性，并可以为各种用户共享。数据库中的数据具有永久存储、有组织、可共享三个基本特点。本节将重点介绍数据库系统中常用的几个基本概念，包括数据、数据处理、数据库和数据库管理系统，以方便读者更好地理解数据库系统的功能和特点。

1.1.1　数据与数据处理

数据（Data）是人们用于记录客观事物情况的符号表示，是表示现实世界中的物体、事件、位置、概念等的未经加工的原始素材，是通过物理观察得来的事实和概念。例如，“四川省内 8 所高校 2009 年本科扩招近 3 000 人”，其中的数据除了“8 所”、“2009 年”和“3 000 人”外，还包括“四川”、“高校”和“本科”。数据的概念在数据处理领域已经被大大拓宽了，其表现形式不仅仅是数字，也可以是文字、图形、图像、声音等。数据也是计算机领域中的术语，属于软件范畴。软件是由程序与数据两部分构成的，数据是计算机软件中程序加工的原料与结果。

信息（Information）是关于现实世界事物的存在方式或运动状态的反映的综合，是数据中所包含的意义，是加工处理后的数据，是数据所表达的内容。具体来说，是一种被加工为特定形式的数据，但这种数据形式对接收者来说是有意义的，而且对当前和将来的决策具有明显的或实际的价值。数据是用来记录信息的可识别的符号，是信息的具体表现形式。描述事物的符号记录称为数据。这些符号可以经过数字化处理后存入计算机。目前数据的概念在数据处理领域中已被大大地拓宽了，因为计算机存储和处理的对象十分广泛。表示这些对象的数据也越来越复杂了。可用多种不同的数据形式表示同一信息，而信息不随数据形式的不同而改变。数据

是信息的符号表示或载体，信息是数据的内涵，是数据的语义解释。数据的表现形式还不能完全表达其内容，不能反映其信息，需要经过解释，数据和数据的解释是不可分的。例如，任意一个数字 19，可以表示一个人的年龄，也可以是 19 号、19 排或其他解释。同样一条数据“王伟，男，1991.06，工商”，可以表达这样一个信息，王伟是一个工商专业的学生，男，出生年月为 1991 年 6 月，也可以对该数据作另外的解释，王伟，男，1991 年 6 月参加工作，是工商专业的老师。数据的解释是对数据含义的说明，数据的含义称为数据的语义，数据与其语义是不可分的。因此，数据是符号化的信息，信息是语义化的数据，如上例中的数据 19 被赋予了特定的语义，就具有了传递信息的功能。一般来说，从信息转换为数据需要进行特征抽取，而从数据还原为信息需要经过数据解释。在一些不是很严格的场合，对信息和数据没有严格的区分，有时甚至相互替换使用，如信息处理与数据处理、信息采集与数据采集等。

数据处理是将数据转换成信息的过程，包括对数据的收集、存储、检索、加工、变换、传输等一系列活动。其基本目的是从大量的、杂乱无章的、难以理解的原始数据中抽取和推导出有价值的信息。可用下式简单地表示数据、数据处理与信息的关系：

数据→数据处理→信息

数据是输入，而信息是输出结果。“数据处理”的真正含义是为了产生信息而处理数据。数据处理的中心问题是数据管理。

1.1.2　数据管理的发展历程

数据库技术是伴随着数据管理任务的需要而产生的，数据处理包括数据的收集、存储、检索、加工、变换、传输等一系列活动，而数据管理则是对数据进行分类、组织、编码、存储、检索和维护。数据管理是数据处理的中心问题。人们研制计算机的初衷就是为了进行复杂的数据运算，但是随着计算机技术的发展，其应用远远超出了这个范围。在计算机软件、硬件的基础上，在需求的推动下，数据管理技术经历了人工管理、文件系统管理和数据库管理（分布式）三个阶段。

1．人工管理阶段

20 世纪 50 年代初，计算机主要用于科学计算，没有操作系统和管理数据的软件，也没有磁盘等直接存取数据的设备，只有纸片、卡片、磁带，数据以符号的形式存放在穿孔卡片上，并以批处理方式运行。数据的组织和管理只能由程序员以人工手段进行，在计算机上无法保存、检索数据。不同程序之间的数据是相互孤立的，逻辑上没有任何联系。人工管理阶段的特点如下。

（1）数据不独立。一组数据对应一组程序，这就使得程序依赖于数据，一旦数据的逻辑结构或者物理结构发生变化，程序就必须做相应的修改，这样就加重了程序员的负担。

（2）数据不保存。当时的计算机主要用于科学计算，数据不需要长期保存。计算某一问题时导入，用完即删除，应用软件与系统软件均如此。

（3）数据非共享。数据是面向应用程序的，一组程序对应一组数据，当多个程序涉及相同的数据时，也无法互相利用、互相参照，必须各自定义，因此程序与程序间存在着大量的冗余数据。

（4）数据无管理。数据由相应的程序管理，没有专门的管理数据的软件，应用程序除了要规定数据的逻辑结构外，还要规定数据的物理结构，包括存储结构、存储方法、输入输出方法，这就给应用程序设计人员增加了很大的负担。

2．文件系统管理阶段

20 世纪 50 年代后期到 60 年代中期，出现了磁盘、磁鼓等直接存储设备。处理方式上不仅有了批处理，而且能够联机实时处理。计算机在一种称为操作系统的软件的指挥下工作，数据以文件的形式存储在计算机上，操作系统中有了专门用于数据管理的软件，一般称为文件系统。

在文件系统中，文件被某一应用所有，数据与程序的独立性差，数据的逻辑结构一旦改变就必须修改应用程序，文件系统只是减轻了程序员对物理设备存取的负担，它并不理解数据的语义，只负责存储，数据的语义信息只能由程序来解释。也就是说，数据收集以后怎么组织，以及数据取出来之后按什么含义应用，只有全权管理它的程序知道。一个应用若想共享另一个应用生成的数据，必须同另一个应用沟通，了解数据的语义与组织方式。在这样的系统中还存在很多弊端。像在学生管理系统中，学校中有教务、学工等诸多部门，分别管理学生的学籍、选课、人事等数据。各个部门使用不同的应用程序，每个应用程序管理一组数据，即数据是面向某个应用的，数据之间是相互隔离的，这致使相同的信息可能在几个地方重复存储，导致数据的冗余度较大，这种冗余除了导致存储和访问开销增大以外，还可能导致数据不一致，即同一数据的不同副本的值不一样。用文件系统管理数据具有如下特点。

（1）数据可以长期保存。这一阶段，计算机不仅用于科学计算，还大量用于信息管理。由于计算机大量用于数据处理，数据需要长期保留在外存上反复进行保存、查询、修改、插入和删除等基础性工作，因此在计算机中长期保存数据成为现实。

（2）由文件系统管理数据。由专门的软件即文件系统进行数据管理，文件系统把数据组织成相互独立的数据文件，利用“按文件名访问，按记录进行存取”的管理技术，对文件进行修改、插入和删除的操作。文件系统实现了记录内的结构性，但整体无结构。程序和数据之间由文件系统提供存取方法进行转换，使应用程序与数据之间有了一定的独立性，程序员可以不必过多地考虑物理细节，而将精力集中于算法。而且数据在存储上的改变不一定反映在程序上，这大大节省了维护程序的工作量。但是，文件系统仍存在一些缺点。

（3）数据共享性差，冗余度大。在文件系统中，一个文件基本上对应于一个应用程序，即文件仍然是面向应用的。当不同的应用程序具有部分相同的数据时，也必须建立各自的文件，而不能共享相同的数据，因此数据的冗余度大，浪费存储空间。同时由于相关数据的重复存储、各自管理，容易造成数据的不一致性，给数据的修改和维护带来了困难。

（4）数据独立性差。文件系统中的文件是为某一特定应用服务的，文件的逻辑结构对该应用程序来说是优化的，因此要想对现有的数据再增加一些新的应用会很困难，系统不容易扩充。一旦数据的逻辑结构改变，必须修改应用程序，修改文件结构的定义。应用程序的改变，如应用程序改用不同的高级语言等，也将引起文件的数据结构的改变。因此数据与程序之间仍缺乏独立性。可见，文件系统仍然是一个不具有弹性的无结构的数据集合，即文件之间是孤立的，不能反映现实世界事物之间的内在联系。

3．数据库管理阶段

20 世纪 60 年代后期，计算机应用于管理的规模更加庞大，数据量急剧增加，硬件技术的发展使计算机联机存取大量数据成为可能。硬件价格下降，而软件价格上升，使开发和维护系统软件的成本增加。计算机上存储的数据量急剧增加，要求计算机能够联机实时处理各种数据，解决多用户、多应用共享数据的需求，文件系统的数据管理方法已无法适应开发应用系统的需要。因此为了解决多用户、多应用的共享数据需求，使数据可以得到更广泛的应用，数据库技

术应运而生。数据库从诞生之日起就迅速发展，从简单到复杂，从单机到网络，与网络通信技术、面向对象程序设计技术、并行计算技术等相互渗透，相互融合，逐渐发展成为完整的系统，称为数据库系统。数据库系统管理数据比文件系统具有明显的优点，从文件系统到数据库系统，是数据管理技术上质的飞跃。用数据库系统管理数据具有如下特点。

（1）数据结构化。在数据库中，数据不再像文件系统中的数据那样从属于某个特定的应用，而是按照某种数据模型组织成为一个结构化的数据整体。它不仅描述了数据本身的特性，即数据内部的结构化，而且描述了数据与数据之间的种种联系，这使数据库系统中实现了整体数据的结构化。

（2）实现数据共享。在数据库系统中，由于数据是面向整个系统的，数据库中的数据实现了按某种数据模型组织为一个结构化的数据，实现了多个应用程序、多个用户甚至多种语言能够共享一个数据库中的数据（目前数据共享已可实现更大范围的共享），所以大大提高了数据的利用率，也提高了工作效率。数据共享是数据库先进性的重要体现。

（3）数据冗余低，易扩充。由于数据可共享，减少了重复的数据存储，节约了存储空间，而且数据的共享还能避免数据之间的不相容性与数据的不一致性。由于数据面向整个系统，是有结构的数据，因此它不仅可以被多个应用共享使用，而且容易增加新的应用，使得数据库系统弹性大，易于扩充，可以适应各种用户的要求。选取整体数据子集或加上一部分数据，便可用于不同的新应用需求。

（4）数据独立性高。数据库的另一个主要特性之一是实际的数据与使用的数据的程序是分离的，即数据与程序相互独立，互不依赖，不会因为一方的改变而改变另一方，这样大大简化了应用程序设计与维护的工作量，同时数据也不会随程序的结束而消失，可以长期保留在系统中。

数据的独立性包括数据的物理独立性和数据的逻辑独立性，它们的实现是由数据库管理系统的二级映像功能来保证的。

（5）数据由 DBMS 统一控制。数据库管理系统（DBMS）是统一控制和管理数据的软件。由于数据库中数据的共享，可能会造成多个用户同时存取数据，甚至会在同一时间存取同一数据，如果不进行数据控制，数据库中的数据就不正确了。为了保证数据的完整性、安全性，数据库管理系统必须具备以下功能。

① 数据的安全性保护。数据的安全性是指每个用户只能按指定方式使用和处理指定数据，DBMS 可保护数据以防止不合法使用造成的数据的泄密和破坏。

② 数据的完整性检查。系统通过设置一些完整性规则以保证数据的正确性、有效性和相容性，从而将数据控制在有效的范围内，或保证数据之间满足一定的关系。

③ 并发控制。当多个用户同时对数据进行修改、存取时，对多用户的并发进程加以控制和协调，防止相互干扰而得到错误的结果或破坏数据库的完整性。

④ 数据库恢复。在计算机系统出现各种故障时，DBMS 可将数据库从错误状态恢复到某一已知的正确状态。

1.1.3　数据库系统

数据库系统在今天的信息社会中有着广泛的应用，它是信息技术的核心。数据库系统一般由数据库、数据库管理系统、数据库管理员、硬件平台及软件平台组成。DBMS 是数据库系统的基础和核心。

1．数据库

顾名思义，数据库是在计算机存储设备上按一定格式存储数据的仓库。科技飞速发展的今天，人们借助计算机和数据库来保存和管理大量的数据，以便更有效地利用这些信息资源。

长期储存在计算机内的、有组织的、可共享的数据集合称为数据库。数据库中的数据按一定的数据模型组织、描述和储存，具有较小的冗余度、较高的数据独立性和易扩展性，并可为各种用户共享。

2．数据库管理系统

数据库管理系统是管理数据库的系统软件，它负责数据库中的数据组织、数据操纵、数据维护，并保护控制数据不受破坏。

3．数据库管理员

数据库管理员（DataBase Administrator，DBA）是一类特殊的用户，他担负着全面管理和控制数据库系统的任务。其主要职责包括数据库设计、数据库维护和改善系统性能，提高系统效率。随着时间的推移，用户的需求会发生改变，DBA 还需对数据库进行较大的改造，包括修改当初的部分设计，即进行数据库的重构造。

4．硬件平台

在数据库系统中，数据量都很大，并且由于数据库管理系统功能的完善使得其自身规模也很大，因此数据库系统对硬件资源的要求较高。这包括要求计算机具有足够大的内存，用于存放操作系统、DBMS 的核心模块、数据缓冲区及应用程序，有足够大和安全性好的磁盘或磁盘阵列等存储设备存放数据，有足够的存储设备做数据备份，同时还要求有较高的通信能力，以提高数据传送能力。

5．软件平台

数据库系统的软件主要包括以下几项。操作系统，它作为基础软件支持 DBMS 运行；数据库管理系统，建立、使用、维护和配置数据库的系统软件；用于开发应用程序的高级语言及其编译系统；以数据库管理系统为核心的应用开发工具；为某个特定应用环境开发的数据库应用系统。

1.1.4　数据模型

在现实世界中有许多模型，这些模型都是对现实世界中某个对象特征的模拟和抽象，如飞机模型、汽车模型就是对现实世界的飞机和汽车的模拟和抽象。数据模型也是一种模型，只不过它是对现实世界里的数据特征的抽象。由于计算机不能直接处理现实世界的具体事物，因此人们必须先把具体事物转换成计算机能够处理的数据，即把现实世界中具体的人、物、活动、概念等用数据模型来抽象、表示和处理，即先进行数字化。这些就需要我们建立一个数据模型。例如，学生管理信息系统，人们通常应了解在该系统中应有哪些数据，这些数据之间有什么联系，以及如何组织这些数据并将其合理地存放在数据库中，以便有效地对其进行管理。

目前的数据库系统均是基于某一数据模型的，数据模型是数据库的核心和基础。因此，我们必须掌握数据模型的基本概念及基本知识。

1．数据模型的三要素及分类

数据模型具有以下 3 个要素。

1）*数据结构*

数据结构用于描述系统的静态特性，即组成数据库的对象类型，研究与数据类型、内容、

性质有关的对象。数据结构包括两方面的内容：一是数据本身，如关系模型中的关系、域、属性等；二是数据之间的联系。

2）数据操作

描述系统的动态特征，即对数据库中的对象允许执行的操作的集合，主要有检索和更新（包括插入、删除、修改）两大类操作。数据模型必须定义这些操作的确切含义、操作符号、操作规则（如优先级），以及实现操作的语言。

3）数据的约束条件

数据的约束条件是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的制约和存储规则，用以限定符合数据模型的数据库状态及状态的变化，以保证数据的正确、有效、一致。此外，数据模型还应该提供定义完整性约束条件的机制。

为了建立数据模型，首先需要对特定的某一现实世界进行抽象，即找出现实世界的特征，然后继续整理并以数据的形式存储在计算机世界的数据库中。这样抽象出的数据模型应满足 3 个方面的要求：一是能比较真实地模拟现实世界；二是容易为人所理解；三是便于在计算机上实现。

但现实世界和计算机世界存在很大的差异，直接将现实世界中的语义映射到计算机世界是非常困难的。要同时满足这 3 方面要求的数据模型目前还很难实现，即能真实模拟现实世界的模型常常无法在计算机中直接实现。所以在数据库的设计中引入一个信息世界作为通向计算机世界的桥梁。

为了把现实世界中的具体事物抽象、组织为某一 DBMS 支持的数据模型，人们常常先将现实世界抽象为信息世界，然后将信息世界再转换为计算机世界。而将现实世界的客观对象转换为信息世界的数据抽象并不依赖于具体的计算机系统，它不是某个 DBMS 支持的数据模型，仅是一个概念级的模型；然后再把概念模型转换为计算机上某一特定的 DBMS 支持的数据模型。

由此，根据不同的应用目的，数据模型被分为两类。一类是概念数据模型（简称概念模型），它是按用户的观点来对数据和信息建模，用于把现实世界抽象到信息世界；还有一类是结构数据模型（简称数据模型），它是按计算机系统的观点对数据建模，用于把信息世界转换到计算机世界。两类数据模型的关系如图 1-1 所示。

从现实世界到信息世界，即建立概念模型，由数据库设计人员完成。从概念模型到数据模型的转换可以由数据库设计人员完成，也可以用专门的数据库设计工具辅助设计人员完成，各数据库管理系统均提供该工具。

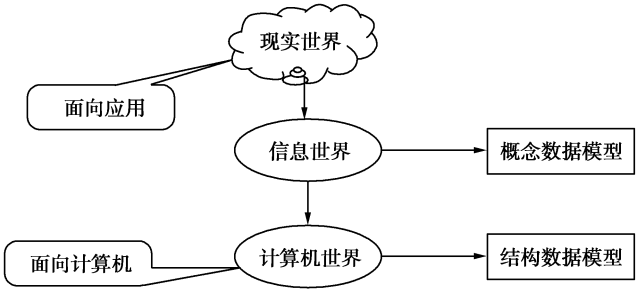


图 1-1　两类数据模型

2．概念模型

概念模型是现实世界到信息世界的第一层抽象，是到计算机世界的一个中间层次，是数据库设计人员进行数据库设计的一个工具，同时也是数据库设计人员和用户之间进行交流的语言，因此就要求概念模型不仅要具有较强的语义表达能力，能够方便、直接地表达应用系统中的各种语义，还应该简单、清晰、易于用户理解。

概念模型只是将现实世界的客观对象抽象为某种信息结构，这种信息结构并不依赖于具体的计算机系统；而对应于计算机世界的模型则由数据模型描述，数据模型是数据库中实体之间及其联系的抽象描述。

数据模型是从计算机实现的观点来对数据建模，一般都有严格的形式化定义，以便于在计算机上实现。常见的数据模型包括层次模型、网状模型、关系模型、面向对象模型及对象关系模型等，其中关系模型是目前最流行的数据库数据模型。

数据模型是严格定义的一组概念的集合。这些概念精确地描述了系统的静态特性、动态特性和完整性约束条件。

实体-联系模型（E-R 模型）是基于对现实世界的这样一种认识而建立的，即世界是由一组称为实体的基本对象及这些对象间的联系组成的。此模型通过允许对应用系统模式进行定义来帮助数据库进行设计，应用系统模式代表了数据库的全局逻辑结构。E-R 模型是一种语义模型，模型的语义方面主要体现在模型力图去表达数据的意义。E-R 模型在将现实世界的含义和相互关联映射到概念模式方面非常有用，因此，许多数据库设计工具都利用了 E-R 模型的概念。

3．实体集

实体（Entity）是现实世界中客观存在并可相互区别的“物体”或“事件”。例如，学校中的每个同学、老师都是一个实体。每个实体有一组特征，其中一部分特征的取值可以唯一标示实体，如学生的身份证号、学号。实体可以是具体的事或物体，也可以是抽象的概念，如银行客户的贷款等。

实体集是具有相同类型及相同性质（或属性）的实体集合。例如，全体学生可被定义为学生实体集（Student），同样，教师实体集（Teacher）表示所有教师信息的集合。

属性（Attribute），实体所具有的某一特征称为属性。属性是实体集中每个成员具有的描述性性质，因此实体可以通过一组属性来表示。学生实体可能有学号、姓名、性别、出生日期、专业、家庭地址等属性，而对每个属性来说，各实体都有自己的属性值。

码（Key），能够唯一标示实体的属性集称为码，如身份证号、学号都可以作为学生实体的码。

域（Domain），每个属性具有一组相同数据类型的值的集合，该集合称为该属性的域或值集，如性别取值为男、女，学生姓名的域是某个长度内的所有字符串的集合。

实体型（Entity Type），一组具有相同属性的实体必然具有共同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体，称为实体型。例如，课程（课程号，课程名，学分）就是一个实体型。

因此数据库包括了一组实体集，每个实体集中包括一些相同类型的实体。在现实世界中，事物与事物之间，以及事物内部都是有联系的，而这些联系反映在信息世界中即表现为实体之间的联系和实体内部的联系。实体之间的联系指不同实体集之间的联系，而实体内部的联系通常指组成实体的各属性之间的联系。

4．实体之间联系的类型

联系是多个实体间的相互关联。例如，定义学生与课程两个实体的关联，这一联系指学生

实体中的某个学生可以选择课程实体中的课程，联系集是同类联系的集合。

两个实体集 A 和 B 之间的联系可以分为 3 类。

1）一对一联系（1∶1）

对于实体集 A 中的一个实体最多同实体集 B 中的一个实体相联系，B 中的一个实体也最多同 A 中的一个实体相联系，则称实体集 A 与实体集 B 是一对一联系，记为 1∶1。

例如，在学生信息管理系统中，一个班级只有一个班长，而一个班长只在一个班中任班长，则班级与班长之间就是一对一联系，如图 1-2 左图所示。

2）一对多联系（1∶n）

对于实体集 A 中的一个实体可以同实体集 B 中的任意数目的实体相联系，而 B 中的一个实体最多只同 A 中的一个实体相联系，则称实体集 A 与实体集 B 是一对多联系，记为 1∶n。

例如，一个辅导员可以管理多个班级，而一个班级只有一个辅导员；一个教研室可以有多个老师，而一个老师只属于一个教研室，则辅导员与班级，教研室与老师之间具有一对多联系，如图 1-2 中图所示。

3）多对多联系（m∶n）

对于实体集 A 中的一个实体可以同实体集 B 中的任意数目的实体相联系，反之，实体集 B 中的一个实体也可以同 A 中的任意数目的实体相联系，则称实体集 A 与实体集 B 具有多对多联系，记为 m∶n。

例如，一个学生可以同时选修多门课程，而一门课程同时有若干个学生选修，则学生与课程之间具有多对多联系，如图 1-2 右图所示。

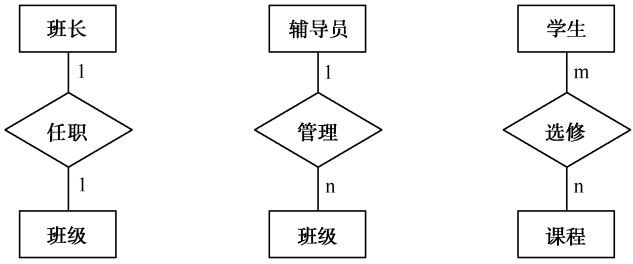


图 1-2　两个实体集之间的联系

在现实中，还存在两个以上的实体集之间的一对一、一对多、多对多联系。这种两个实体集以上的联系可以清晰地表示出几个实体集参与到一个联系集中，而在对应的二元联系中，难以体现这样的参与性约束。

例如，一个学生可以同时选修多门课程，一个老师在教一门课时对应多个学生，同时一门课程可以有若干个学生选修，一个老师可以教多门课，如图 1-3 左图所示。

另一个例子是工厂中常见的供应商、项目、零件 3 个实体集之间的多对多联系。一个供应商可以给多个项目供应多种零件，每个项目可以使用多个供应商供应的多种零件，每种零件可由多个供应商供给，则供应商、项目、零件三者之间是多对多的联系，如图 1-3 右图所示。

同样，在同一个实体集内部的各个实体之间也可以存在一对一、一对多、多对多联系。比如，教师实体集内部即存在领导与被领导的联系。

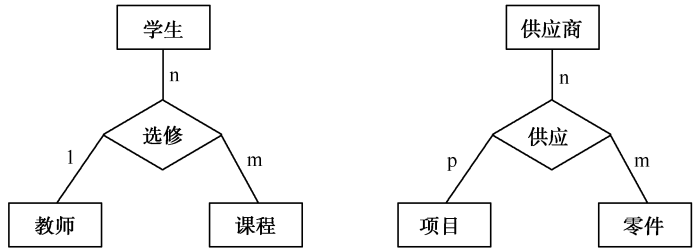


图 1-3　3 个实体集之间的联系

5．实体–联系图

概念模型的表示方法很多，这里我们介绍最常用的实体–联系方法（Entity-Relationship Approach），该方法是 P.P.S.Chen 于 1976 年提出的。该方法用 E-R 图来描述现实世界的概念模型，E-R 方法也称为 E-R 模型。

E-R 图以图形化的方式表示实体及其联系的关系，简单、清晰，易于与用户沟通。E-R 图包括以下几个主要构件。

（1）矩形。表示实体集，矩形内写明实体名。

（2）椭圆。表示属性，椭圆内写明属性名，并用无向线段将其与所属的实体连接起来，如图 1-4 所示。

（3）菱形。表示联系，菱形内标明联系名，并用无向线段将其与有关的实体连接起来，同时应在无向边上注明联系的类型（1∶1，1∶n，n∶m）。

如果联系本身有属性，那么这些属性也要用无向边与该联系连接起来。例如，在学生实体与课程实体的多对多联系中，成绩即是联系的属性，如图 1-5 所示。

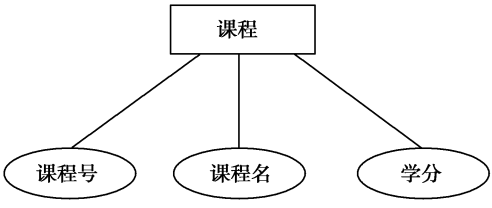


图 1-4　课程实体及其属性

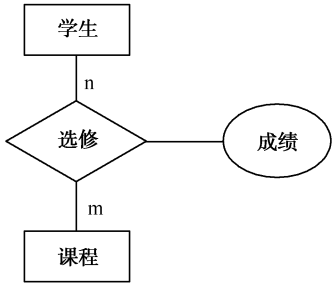


图 1-5　联系的属性

1.1.5　逻辑结构设计

概念结构设计所得的 E-R 模型是对用户需求的一种抽象的表达形式，它独立于任何一种具体的数据模型，因而也不能被任何一个具体的 DBMS 所支持。为了能够建立起最终的物理系统，还需要将概念结构进一步转化为某一 DBMS 所支持的数据模型，然后根据逻辑设计的准则、数据的语义约束和规范化理论等对数据模型进行适当的调整和优化，形成合理的全局逻辑结构，并设计出用户子模式。这就是数据库逻辑结构设计所要完成的任务。

数据库逻辑结构的设计分为两个步骤，首先将概念设计所得的 E-R 图转换为关系模型，然后对关系模型进行优化，如图 1-6 所示。

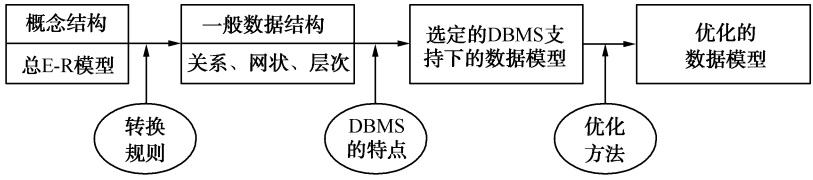


图 1-6　逻辑结构设计的过程

关系模型是一组关系（二维表）的结合，而 E-R 模型则是由实体、实体的属性、实体间的关系 3 个要素组成的。所以要将 E-R 模型转换为关系模型，就是要将实体、属性和联系都转换为相应的关系模型。下面具体介绍转换的规则。

1．一种实体类型转换为一个关系模型

将每种实体类型转换为一个关系模型时，实体的属性就是关系的属性，实体的关键字就是关系的关键字。例如，可将“学生”实体转换为一个关系模型，如图 1-7 所示。其中，带下画线的属性为主属性，该主属性为关系模型的外键。

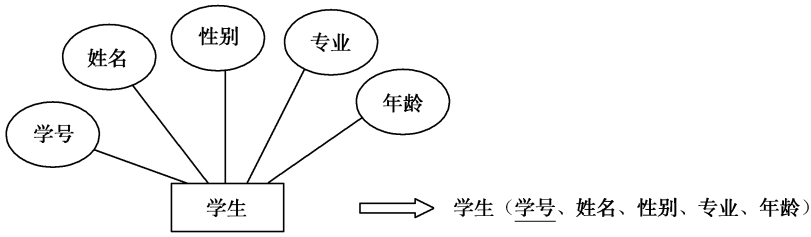


图 1-7　一种实体类型转换为一个关系模型

2．一对一关系（1：1）的转换

一对一关系有以下两种转换方式。

（1）转换为一个独立的关系模型。联系名为关系模型名，与该联系相连的两个实体的关键字及联系本身的属性为关系模型的属性，其中每个实体的关键字均是该关系模型的候选键。

（2）与任意一端的关系模型合并。可将相关的两个实体分别转换为两个关系，并在任意一个关系的属性中加入另一个关系的主关键字。

例如，若某工厂的每个仓库只配备了一名管理员，那么仓库实体与管理员实体间便为 1：1 关系。根据以上介绍的原则，可以进行如图 1-8 所示的变换。

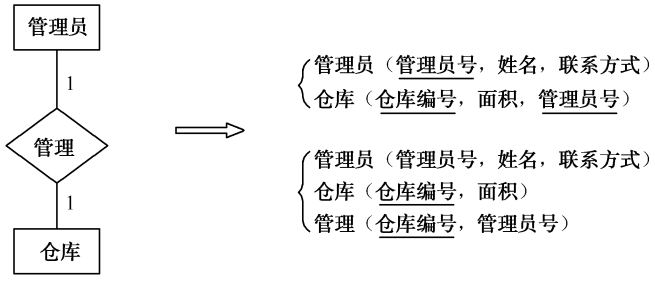


图 1-8　1：1 关系的转换

在实际设计中究竟采用哪种方案可视具体的应用而定。如果经常要在查询仓库关系的同时

查询此仓库管理员的信息，就可选用前一种关系模型，以减少查询时的链接操作。反之，如果在查询管理员时要频繁查询仓库信息，则选用后一种关系模型。总之，在模型转换出现较多方案时，效率是重要的取舍因素。

3．一对多关系（1：n）的转换

一对多关系也有两种转换方式。

（1）将 1：n 关系转换为一个独立的关系模型。联系名为关系模型名，与该联系相连的各实体的关键字及联系本身的属性为关系模型的属性，关系模型的关键字为 n 端实体的关键字。

（2）将 1：n 联系与 n 端关系合并。1 端的关键字及联系的属性并入 n 端的关系模型即可。

如图 1-9 所示，实体“专业”和“学生”之间的联系为 1：n，则两者可使用以上的原则进行关系模型的转换。

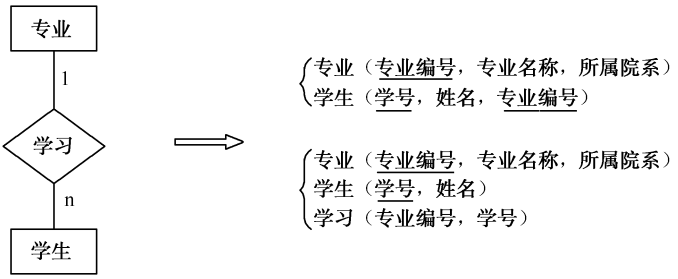


图 1-9　1：n 联系的转换

4．多对多关系（m：n）的转换

关系模型名为关系名，与该关系相连的各实体的关键字及关系本身的属性为关系模型的属性，关系模型的关键字为关系中各实体关键字的并集。

例如，在学校中，一名学生可以选修多门课程，一门课程也可被多名学生选修，则实体“学生”与“课程”之间满足多对多的关系，其转换方法如图 1-10 所示。

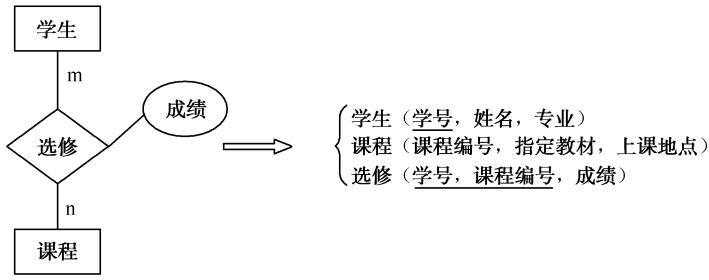


图 1-10　m：n 关系的转换

在设计好一个项目的关系模式后，就可以在数据库管理系统的环境下，创建数据库、关系表及其他数据库对象了，然后输入相应的数据，并根据需要对数据进行各种操作。

1.2　SQL Server 2008 简介

SQL Server 2008 是由 Microsoft 公司发布的最新关系数据库管理系统，它基于 SQL Server 2005 的强大功能，为用户提供了完整的数据管理和分析解决方案。

SQL Server 2008 在 Microsoft 的数据平台上发布，可帮助用户的组织随时随地管理任何数据。它可以将结构化、半结构化和非结构化文档的数据（如图像和音乐）直接存储到数据库中。SQL Server 2008 提供了一系列丰富的集成服务，可以对数据进行查询、搜索、同步、报告和分析之类的操作。数据可以存储在各种设备上，从数据中心最大的服务器一直到桌面计算机和移动设备，用户可以控制数据而不用管数据存储在哪里。

SQL Server 2008 允许用户在使用 Microsoft .NET 和 Visual Studio 开发的自定义应用程序中使用数据，以及在面向服务的架构（SOA）和通过 Microsoft BizTalk Server 进行的业务流程中使用数据。信息工作人员可以通过他们日常使用的工具（如 2007 Microsoft Office 系统）直接访问数据。SQL Server 2008 提供了一个可信的、高效率的智能数据平台，可以满足用户的所有数据需求。

SQL Server 2008 的特点如下。

1．可信

SQL Server 为用户的业务关键型应用程序提供最高级别的安全性、可靠性和伸缩性。

1）保护有价值的信息

透明的数据加密。允许加密整个数据库、数据文件或日志文件，无须更改应用程序。这样做的好处包括，可以同时使用范围和模糊搜索来搜索加密的数据，未经授权的用户只能搜索安全的数据，可以在不更改现有应用程序的情况下进行数据加密。

可扩展的键管理。SQL Server 2008 为加密和键管理提供了一个全面的解决方案。它可以支持第三方键管理和 HSM 产品，以满足不断增长的需求。

审计。SQL Server 2008 通过 DDL 创建和管理审计，同时通过提供更全面的数据审计来简化遵从性。这允许组织回答常见的问题，如“检索什么数据”。

2）确保业务连续性

增强的数据库镜像。SQL Server 2008 构建于 SQL Server 2005 之上，但增强的数据库镜像包括自动页修复、提高性能和提高支持能力，因而是一个更加可靠的平台。

数据页的自动恢复。SQL Server 2008 允许主机和镜像机器从 823/824 类型的数据页错误中透明地恢复，它可以从透明于终端用户和应用程序的镜像伙伴处请求新副本。

日志流压缩。数据库镜像需要在镜像实现的参与方之间进行数据传输。使用 SQL Server 2008 可为参与方之间的输出日志流压缩提供最佳性能，并最小化数据库镜像使用的网络带宽。

3）启用可预测的响应

资源管理者。SQL Server 2008 通过引入资源管理者来提供一致且可预测的响应，允许组织为不同的工作负荷定义资源限制和优先级，这允许并发工作负荷为它们的终端用户提供一致的性能。

可预测的查询性能。SQL Server 2008 通过提供功能锁定查询计划来支持更高的查询在性能上的稳定性和可预测性，允许组织在硬件服务器替换、服务器升级和生产部署之间推进稳定的查询计划。

数据压缩。它可更有效地存储数据，并减少数据的存储需求。数据压缩还为大量 I/O 边界工作（如数据仓库）提供极大的性能提高。

热添加 CPU。允许 CPU 资源在支持的硬件平台上添加到 SQL Server 2008，以动态调节数据库大小而不强制应用程序死机。注意，SQL Server 已经可以支持在线添加内存资源的能力了。

2．高效率

为了抓住如今风云变幻的商业机会，公司需要能力来快速创建和部署数据驱动的解决方案。SQL Server 2008 减少了管理和开发应用程序的时间和成本。

1）按照策略进行管理

Policy-Based Management。它是一个基于策略的系统，用于管理 SQL Server 2008 的一个或多个实例。将其与 SQL Server Management Studio 一起使用可以创建管理服务器实体（如 SQL Server 实例、数据库和其他 SQL Server 对象）的策略。

精简的安装。SQL Server 2008 通过重新设计安装、设置和配置体系结构，对 SQL Server 服务生命周期进行了巨大的改进。这些改进将在硬件上的物理安装与 SQL Server 软件的配置隔离，允许组织和软件合作伙伴提供推荐的安装配置。

性能数据收集。性能调节和故障诊断对于管理员来说是一项耗时的任务。为了给管理员提供可操作的性能检查，SQL Server 2008 包含更多详尽性能数据的集合，如一个用于存储性能数据的集中化的新数据仓库，以及用于报告和监视的新工具。

2）简化应用程序开发

语言集成查询（LINQ）。开发人员可以使用诸如 C#或 VB.NET 等托管的编程语言而不是 SQL 语句查询数据。允许使用根据 ADO.NET（LINQ to SQL）、ADO.NET DataSets（LINQ to DataSet）、ADO.NET Entity Framework（LINQ to Entities），以及实体数据服务映射供应商运行.NET 语言编写的无缝、强类型、面向集合的查询。新的 LINQ to SQL 供应商允许开发人员在 SQL Server 2008 表和列上直接使用 LINQ。

ADO.NET Object Services。ADO.NET 的 Object Services 层将具体化、更改跟踪和数据持久作为 CLR 对象。使用 ADO.NET 框架的开发人员可以使用 ADO.NET 管理的 CLR 对象进行数据库编程。SQL Server 2008 引入更有效、优化的支持来提高性能和简化开发。

3）存储任何信息

DATE/TIME。SQL Server 2008 引入的新的日期和时间数据类型。DATE 仅表示日期的类型。TIME 仅表示时间的类型。DATETIMEOFFSET 可以感知时区的 DATETIME 类型。DATETIME2 比现有 DATETIME 类型具有更长数位和更大年份范围。新的数据类型允许应用程序拥有独立的日期和时间类型，同时为时间值提供更大的数据范围或用户定义的精度。

HIERARCHY ID。允许数据库应用程序使用比当前更有效的方法来制定树结构的模型。新的系统类型 HIERARCHY ID 可以存储代表层次结构树中节点的值。这种新类型将作为一种 CLR UDT 实现，它将暴露几种有效并有用的内置方法，用于使用灵活的编程模型创建和操作层次结构节点。

FILESTREAM Data。允许大型二进制数据直接存储在 NTFS 文件系统中，同时保留数据库的主要部分并维持事务一致性。允许扩充传统上由数据库管理的大型二进制数据，可以存储在数据库外部的更加经济的存储设备上，而没有泄密风险。

集成的全文本搜索。它可使文本搜索和关系型数据之间进行无缝转换，同时允许用户使用文本索引在大型文本列上执行高速文本搜索。

Sparse Columns。NULL 数据不占据物理空间，它提供了高效的方法来管理数据库中的空数据。例如，Sparse Columns 通常允许有许多空值的对象模型存储在 SQL Server 2005 数据库中，而无须耗费大量空间成本。

大型用户定义的类型。SQL Server 2008 消除了用户定义类型（UDT）的 8 KB 限制，允许

用户极大地扩展其 UDT 的大小。

空间数据类型。通过使用对空间数据的支持，将空间能力构建到用户的应用程序中。例如，使用地理数据类型实现“圆面地球”解决方案，使用经纬度来定义地球表面的区域，使用地理数据类型实现“平面地球”解决方案。存储与投影平面表面和自然平面数据关联的多边形、点和线，如内部空间。

3．智能

SQL Server 2008 提供全面的平台，在用户需要的时候提供智能化服务。

1）集成任何数据

备份压缩。在线保存基于磁盘的备份昂贵且耗时。借助 SQL Server 2008 备份压缩，在线保存备份所需的存储空间更少，备份运行速度更快，因为需要的磁盘 I/O 更少。

已分区表并行。分区允许组织更有效地管理增长迅速的表，可以将这些表透明地分成若干易于管理的数据块。SQL Server 2008 继承了 SQL Server 2005 中的分区优势，但提高了大型分区表的性能。

星型连接查询优化。SQL Server 2008 为常见的数据仓库场景提供了改进的查询性能。星型连接查询优化通过识别数据仓库连接模式来减少查询响应时间。

Grouping Sets。它是对 GROUP BY 子句的扩展，允许用户在同一个查询中定义多个分组。Grouping Sets 生成单个结果集（等价于不同分组行的一个 UNION ALL），使得聚集查询和报告变得更加简单快速。

更改数据捕获。使用“更改数据捕获”，可以捕获更改内容并存放在更改表中。它能捕获完整的更改内容，维护表的一致性，甚至还能捕获跨模式的更改。这使得组织可以将最新的信息集成到数据仓库中。

MERGE SQL 语句。随着 MERGE SQL 语句的引入，开发人员可以更加高效地处理常见的数据仓库存储应用场景，如检查某行是否存在，然后执行插入或更新操作。

SQL Server Integration Services（SSIS）管道线改进。“数据集成包”现在可以更有效地扩展，可以利用可用资源和管理最大的企业规模工作负载。新的设计将运行时的伸缩能力提高到多个处理器。

SQL Server Integration Services（SSIS）持久查找。执行查找的需求是最常见的 ETL 操作之一。这在数据仓库中特别普遍，其中事实记录需要使用查找将企业关键字转换成相应的替代字。SSIS 增强查找的性能以支持最大的表。

2）发布相关的信息

分析规模和性能。SQL Server 2008 使用增强的分析能力和更复杂的计算及聚集交付更广泛的分析。新的立方体设计工具帮助用户精简分析基础设施的开发，让他们能够为优化的性能构建解决方案。

块计算。块计算在处理性能方面有了极大的改进，允许用户增加其层次结构的深度和计算的复杂性。

写回。新的 MOLAP 在 SQL Server 2008 Analysis Services 中启用写回（writeback）功能，不再需要查询 ROLAP 分区。这可为用户提供分析应用程序中增强的写回场景，而不牺牲传统的 OLAP 性能。

3）推动可操作的商务洞察力

企业报表引擎。报表可以使用简化的部署和配置在组织中方便地分发（内部和外部）。这

使得用户可以方便地创建和共享任何规格和复杂度的报表。

Internet 报表部署。通过在 Internet 上部署报表，可以很容易地找到客户和供应商。

管理报表体系结构。通过集中化存储和所有配置设置的 API，使用内存管理、基础设施巩固和更简单的配置来增强支持能力和控制服务器行为的能力。

Report Builder 增强。通过报表设计器轻松构建任何结构的特殊报表和创作报表。

内置的表单认证。内置的表单认证让用户可以在 Windows 和 Forms 之间方便地切换。

报表服务器应用程序嵌入。报表服务器应用程序嵌入使得报表和订阅中的 URL 可以重新指向前端应用程序。

Microsoft Office 集成。SQL Server 2008 提供新的 Word 渲染，允许用户通过 Microsoft Office Word 直接使用报表。此外，现有的 Excel 渲染器已经得到极大的增强，以支持嵌套的数据区域、子报表以及合并的表格改进等功能。

预测性分析。SQL Server Analysis Services 继续交付高级的数据挖掘技术。更好的时间序列支持增强了预测能力。增强的挖掘结构提供更大的灵活性，可以通过过滤执行集中分析，还可以提供超出挖掘模型范围的完整信息报表。新的交叉验证允许同时确认可信结果的精确性和稳定性。此外，针对 Office 2007 的 SQL Server 2008 数据挖掘附件提供的新特性使组织中的每个用户都可以在桌面上获得更多可操作的洞察。

总之，SQL Server 2008 为公司和企业提供了可依靠的技术和能力来接受对于管理数据和给用户发送信息的挑战。它是一个可信任的、高效的和智能的数据平台。许多新推出的特性和关键的改进，使它成为迄今为止最强大和最全面的 SQL Server 版本。

1.3　SQL Server 2008 的安装

本书以 SQL Server Express 版本为例介绍 SQL Server 2008 的安装过程。在开始实际安装 SQL Server 2008 之前，首先应确定运行 SQL Server 2008 的计算机的硬件配置要求，其次还应了解 SQL Server 2008 可运行的操作系统版本及特点，最后值得一提的是，在安装 SQL Server 2008 之前，一定要卸载之前的旧版本。

SQL Server Express 的所有版本均可免费下载，并可在协议的许可下重新分发。每一版本既可充当客户端数据库，又可充当基本服务器数据库。任意一版 SQL Server Express 都是独立软件供应商（ISV）、服务器用户、非专业开发人员、Web 开发人员、网站主人，以及创建客户端应用程序的业余爱好者的理想之选。如果您需要使用更高级的数据库功能，则可以将 SQL Server Express 升级到更复杂的 SQL Server 版本。

1.3.1　SQL Server 2008 的环境需求

SQL Server 2008 Express 在系统配置规模上拥有以下的局限性：只能用于一个 CPU，最大内存为 1GB，数据库大小不能超过 4GB。

SQL Server 2008 Express 是专门为小规模服务器和台式机而设计的，因此可使用以下的系统配置：内存至少 512MB；硬盘至少有 600MB 可用空间；CPU 为 Pentium III 1 GHz 或更高级；操作系统为 Windows Server（任何版本）、Windows XP 或 Windows Vista；附加软件为.NET Framework、Windows Installer 1.0 和 Internet Explorer 6.0 SP1 或更新的版本。在安装.NET Framework 时需要重新启动操作系统。如果安装 Windows Installer 也需要重新启动操作系统，

则安装程序将等到.NET Framework 和 Windows Installer 组件安装完成后，再重新启动。

针对日常开发，使用 SQL Server 2008 Express 就足够了。与企业版比起来 Express 版本是受限制的，如数据库最大使用 1GB 内存、数据库文件最大尺寸为 4GB 和使用 1 个 CPU 等，即便是计算机有更多的 CPU、内存和磁盘空间。但是这对于开发和在小型网站上使用已经足够了。因为是免费的，所以没有侵犯版权的担心。

1.3.2　SQL Server 2008 的安装步骤

从光盘或网络获取 SQL Server 2008 的安装光盘，然后就可以安装了。下面以在 Windows XP 平台上安装 SQL Server 2008 为例，安装步骤如下。

（1）将安装盘放入光驱，此时会自动播放并打开安装程序的导航界面，若没有打开也可以直接双击“光盘\Servers\splash.hta”文件来运行。

（2）从导航界面的【安装】区域中单击【服务器组件、工具、联机丛书和示例】链接来启动安装程序，若上一步没有执行也可以直接运行“光盘\Servers\Setup.exe”文件。

（3）通过前面的介绍可知，SQL Server 2008 需要.NET Framework 3.5 版本的支持。因此，安装启动后首先测试是否有.NET Framework 3.5 环境。如果没有会弹出安装对话框，通过启用复选框来接受.NET Framework 3.5 许可协议，再单击【下一步】按钮进行安装，当.NET Framework 3.5 安装完成后单击【完成】按钮。

（4）系统弹出 SQL Server 2008 安装过程的第一个对话框，如图 1-11 所示。单击【安装】按钮，启动【全新 SQL Server 独立安装或向现有安装添加功能】选项。

图 1-11　SQL Server 安装中心

（5）启动【全新 SQL Server 独立安装或向现有安装添加功能】选项之后，系统会弹出安装程序支持规则界面，如图 1-12 所示。系统配置检查器将验证要运行安装的计算机。在此版本中，所做检查包括以下几项。

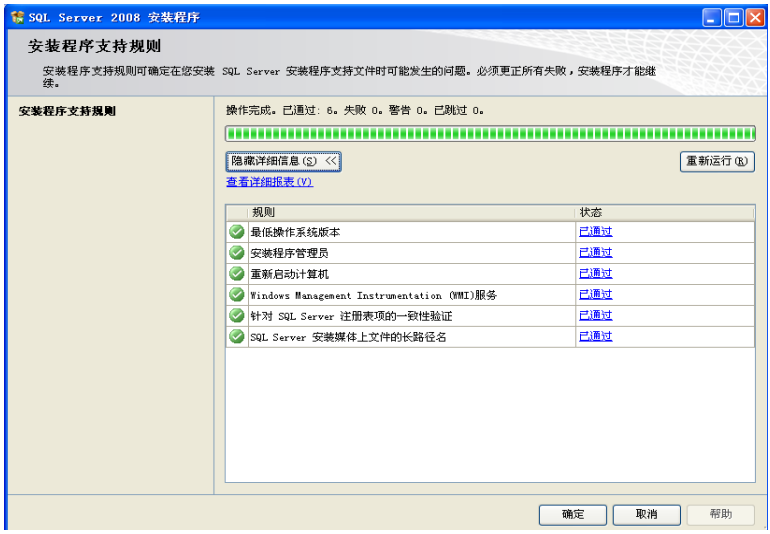


图 1-12　安装程序支持规则（一）

- 操作系统版本检查：验证操作系统是否支持此版本。
- 重新启动要求检查：验证是否有锁定的文件或进程会阻止 SQL Server 安装程序。
- WMI 服务检查验证：Windows Installer 服务是否正常运行。
- 性能计数器一致性检查：检查注册表项的值以验证 SQL Server Performance 计数器安装的增量是否正确。
- Business Intelligence Development Studio 检查：验证是否安装了 Business Intelligence Development Studio，因为不支持此组件的升级。
- 检查以前的 SQL Server 2008 的安装：验证运行安装程序的计算机上是否有 SQL Server 2008 CIP 安装。

（6）待所有检查项都通过验证后，【下一步】按钮被激活。单击它继续安装，如图 1-13 所示。

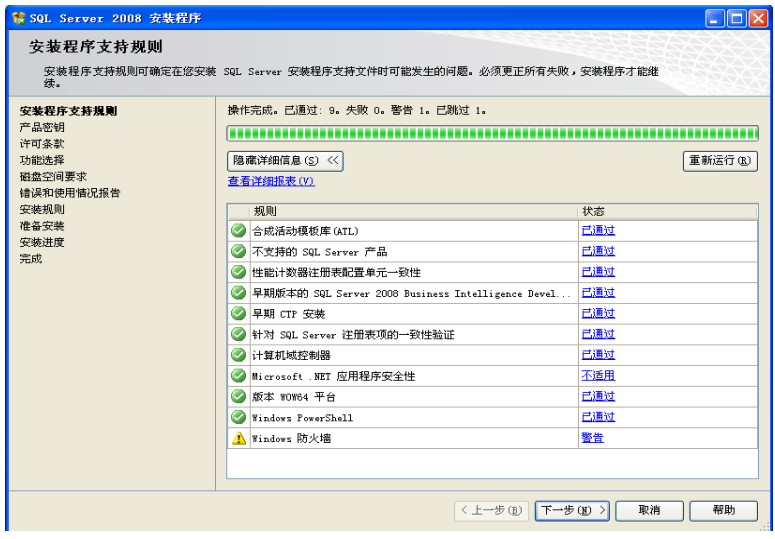


图 1-13　安装程序支持规则（二）

（7）单击【下一步】按钮，显示了要安装 SQL Server 2008 必须接受的软件许可条款。启用【我接受许可条款】复选框后，单击【下一步】按钮继续安装，如图 1-14 所示。

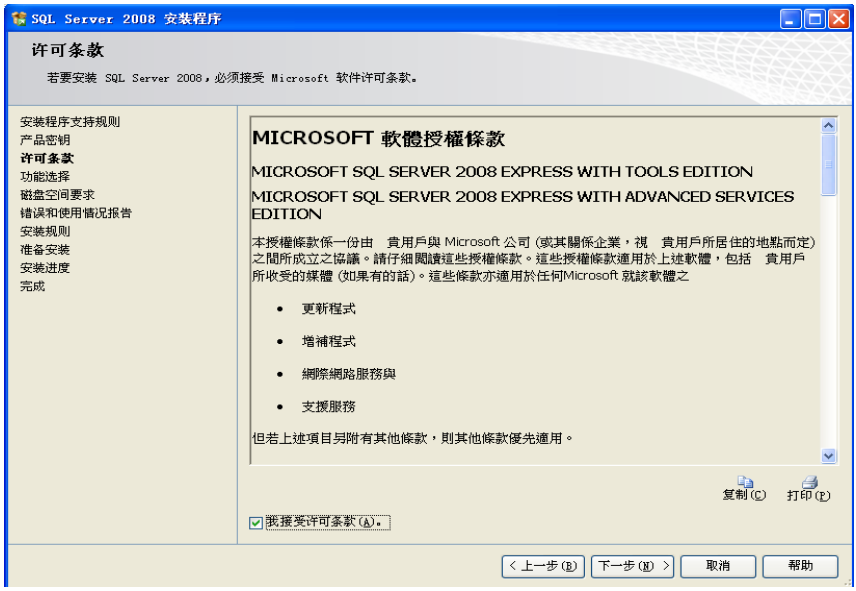


图 1-14　许可条款

（8）接受许可条款之后，系统会检测计算机上是否安装有 SQL Server 必备组件，若没有，安装向导将安装它们。这些必备组件包括.NET Framework 3.5、SQL Server Native Client 和 SQL Server 安装程序的支持文件，单击【安装】按钮开始安装，如图 1-15 所示。

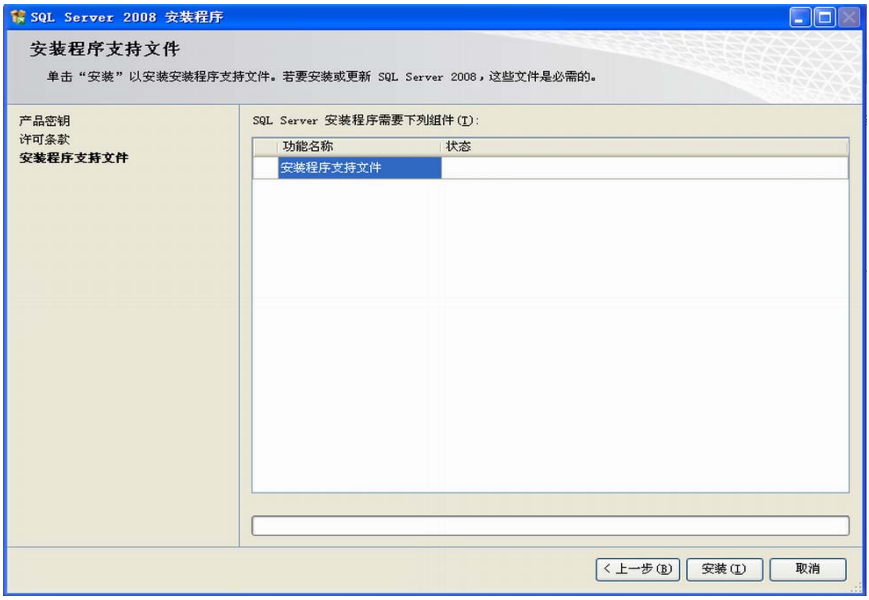


图 1-15　安装程序支持文件

（9）进入功能选择界面，从【功能】区域选择要安装的组件。在启用功能名称复选框后，右侧窗口中会显示每个组件的说明。用户可以选中任意复选框，这里为全选，如图 1-16 所示。

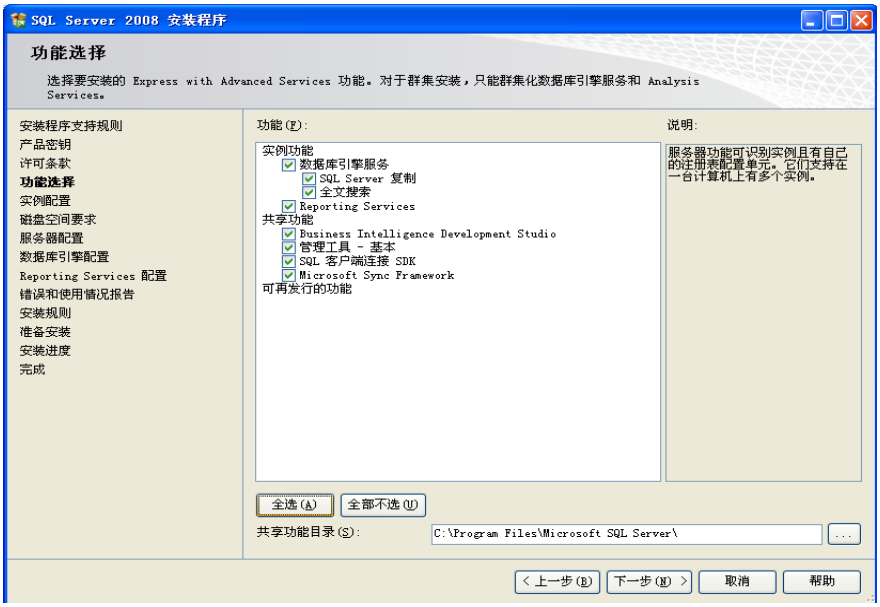


图 1-16　功能选择

（10）单击【下一步】按钮，指定要安装【默认实例】还是【命名实例】。本例指定【默认实例】，如果选择【命名实例】还需指定实例名称，如图 1-17 所示。

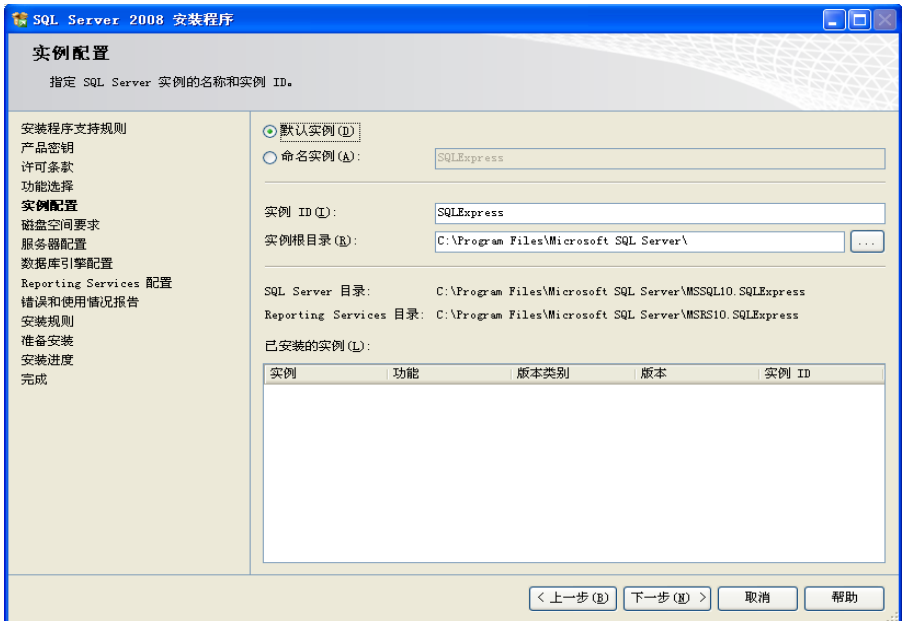


图 1-17　实例配置

（11）继续单击【下一步】按钮，安装程序将检查磁盘的可用空间，如图 1-18 所示。

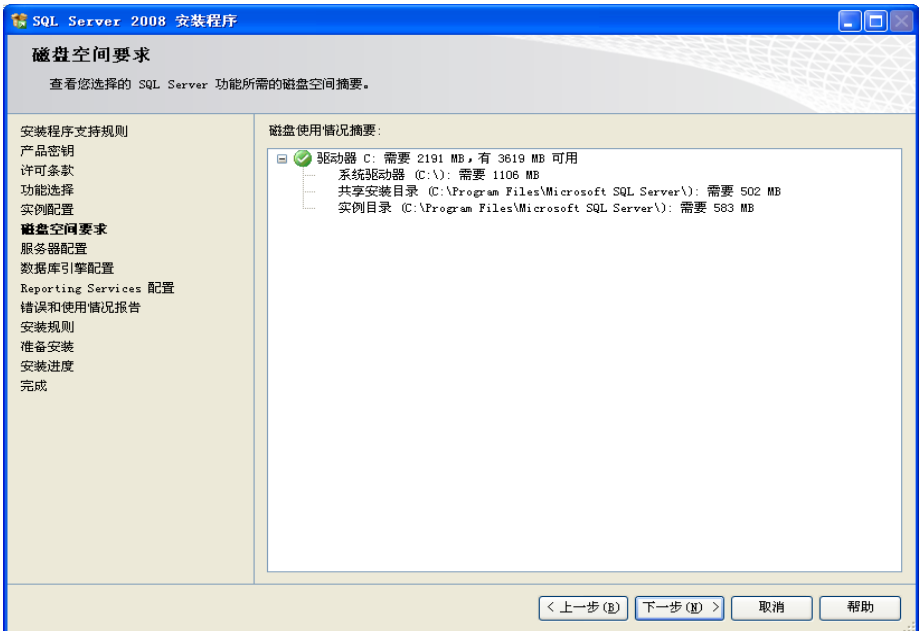


图 1-18　磁盘空间要求

（12）单击【下一步】按钮进入服务器配置界面，单击【服务账户】选项卡，为每个 SQL Server 服务单独配置用户名、密码和启动类型，如图 1-19 所示。

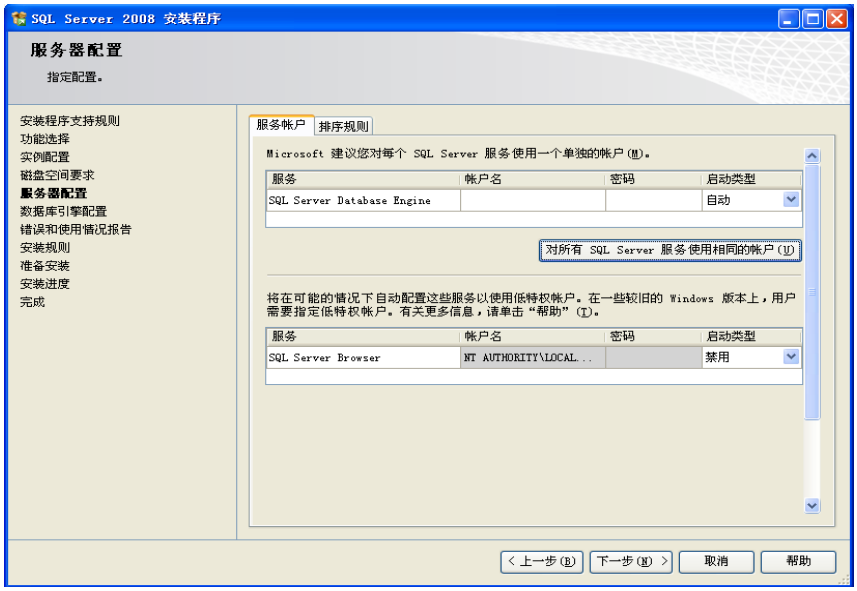


图 1-19　服务账户

（13）配置完服务账户，再单击【排序规则】选项卡，为【数据库引擎】和【Analysis Services】指定非默认的排序规则，如图 1-20 所示。默认情况下，会选定针对英语系统区域设置的 SQL 的排序规则。非英语区域设置的默认排序规则由用户计算机的 Windows 系统区域设置。

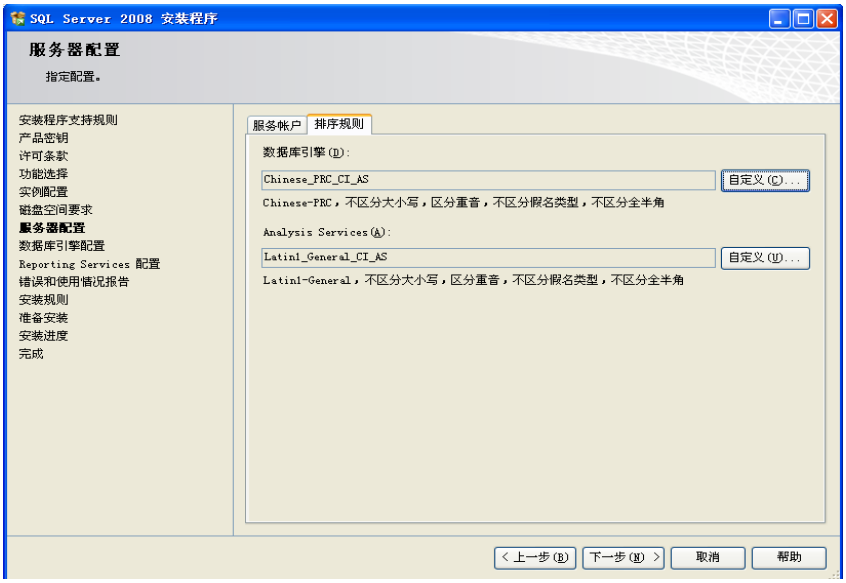


图 1-20　排序规则

（14）单击【下一步】按钮对 SQL Server 2008 的数据库引擎进行配置，单击【账户设置】选项卡，如图 1-21 所示。SQL Server 提供了 Windows 模式和混合模式两种身份验证模式，如果选择混合模式身份验证，就必须为内置 SQL Server 系统管理员账户提供一个强度较高的密码并进行确认。在【指定 SQL Server 管理员】中，必须至少为 SQL Server 实例指定一个系统管理员。若要添加用以运行 SQL Server 安装程序的账户，可以单击【添加当前用户】按钮。若要向系统管理员列表中添加账户或从中删除账户，单击【添加】或【删除】按钮，然后编辑将对其分配 SQL Server 实例的管理员权限的用户、组或计算机的列表。

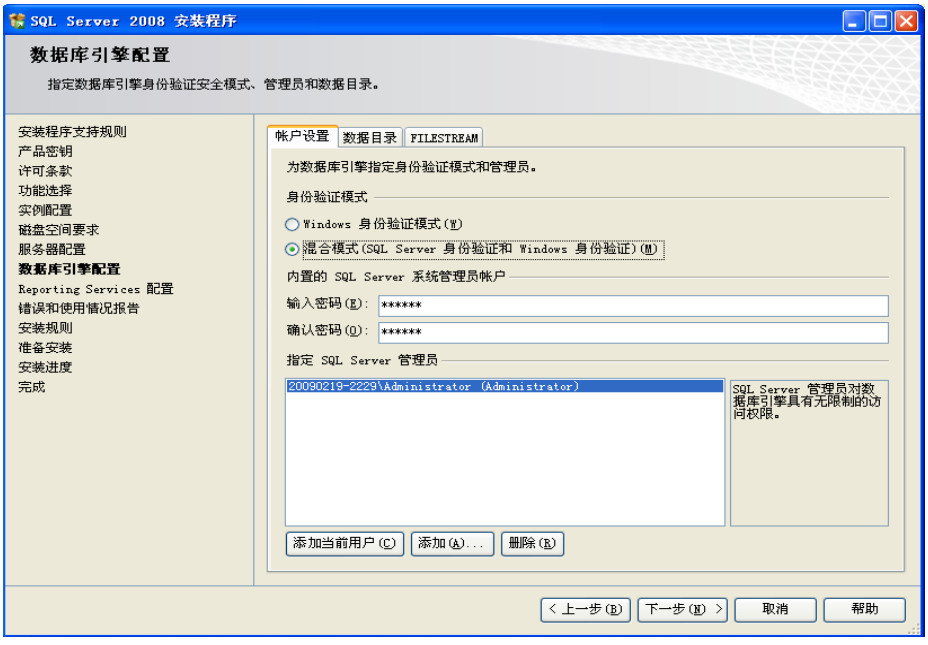


图 1-21　数据库引擎配置（账户设置）

（15）账户设置完成后，单击【数据目录】选项卡，在这里指定各种数据库的安装目录及备份目录，可以使用默认的安装目录，直接单击【下一步】按钮，如图 1-22 所示。

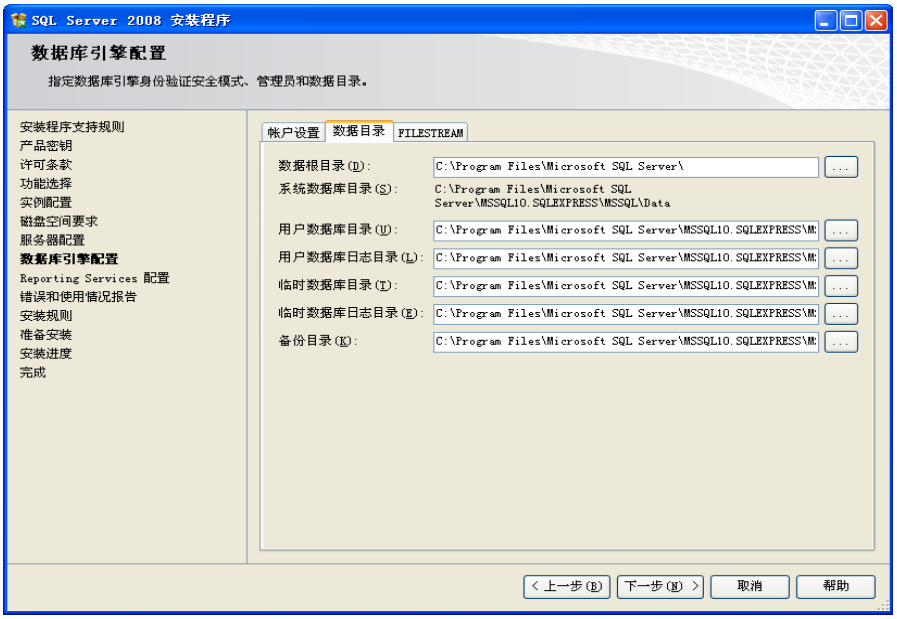


图 1-22　数据库引擎配置（数据目录）

（16）单击【FILESTREAM】选项卡，启用针对 Transact-SQL 的 FILESTREAM 功能，如图 1-23 所示单击【下一步】按钮继续安装。

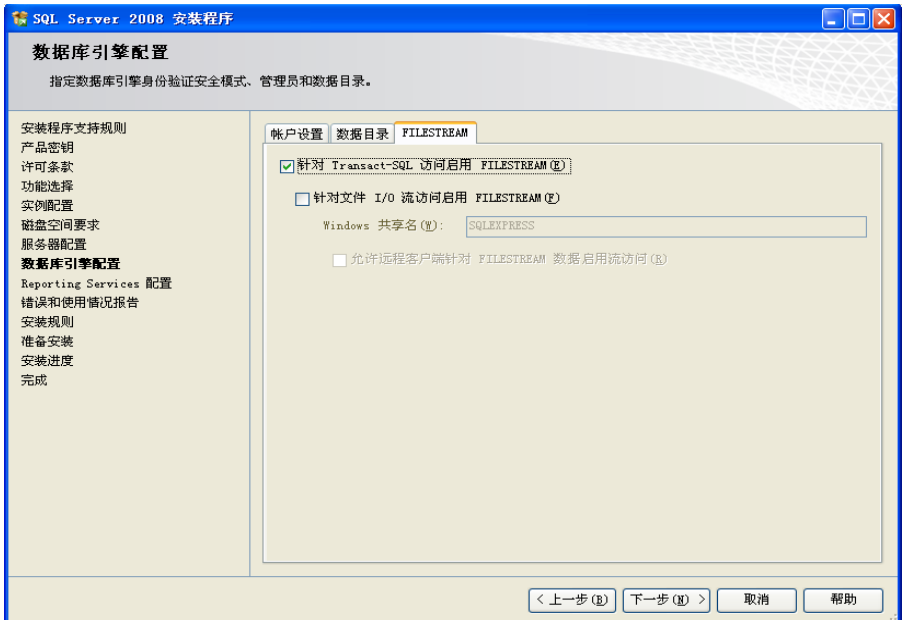


图 1-23　数据库引擎配置（FILESTREAM）

（17）完成数据库引擎配置后，弹出 Reporting Services 的配置界面，这里使用默认配置，如图 1-24 所示。

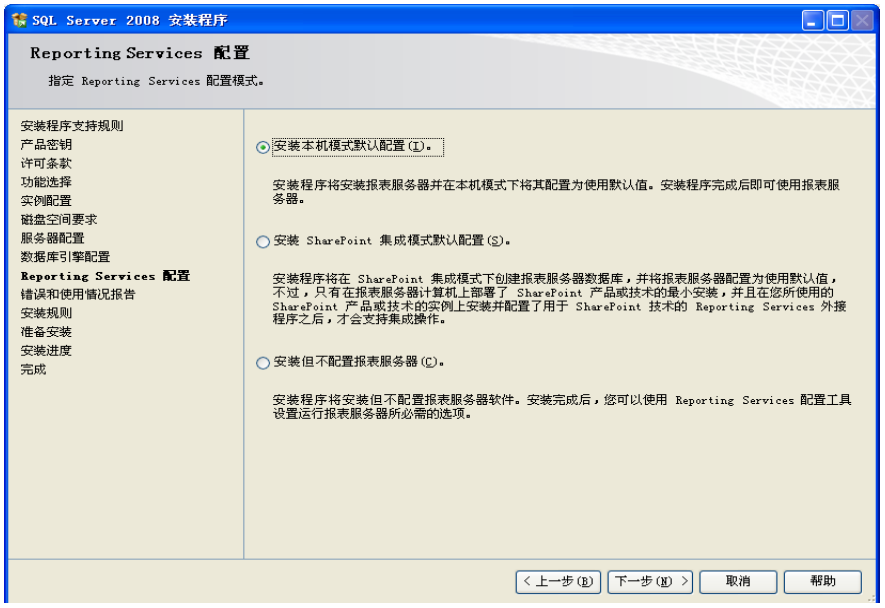


图 1-24　Reporting Serices 配置

（18）单击【下一步】按钮对 SQL Server 2008 的错误和使用情况报告进行设置，通过启用复选框来选择某些功能，如图 1-25 所示。

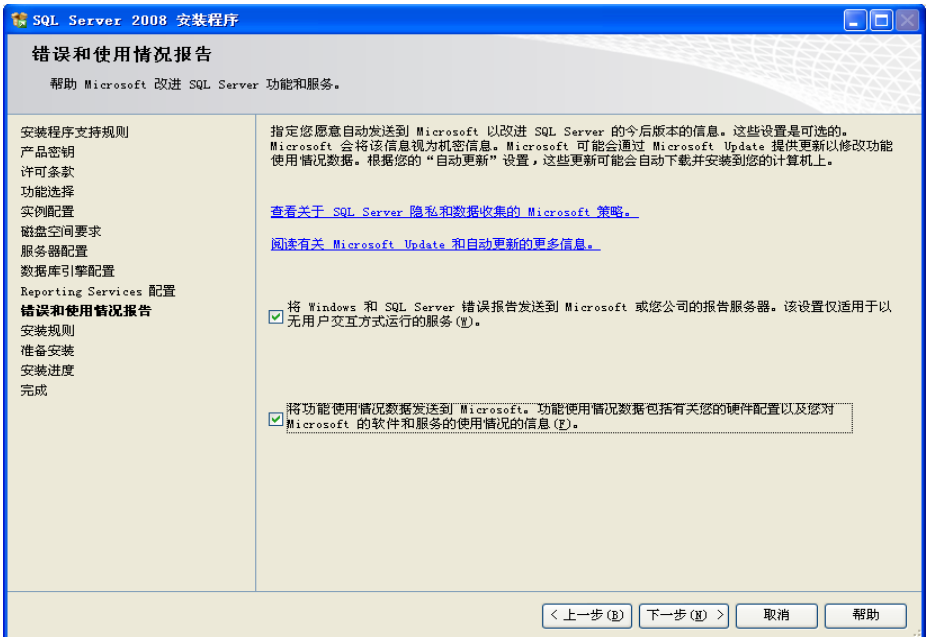


图 1-25　错误和使用情况报告

（19）单击【下一步】按钮结束对 SQL Server 2008 的安装所需参数的配置，进入准备安装界面。在该界面的列表框中，显示了所有要安装的组件，用户可以通过扩展/折叠来查看详细信息，如图 1-26 所示。

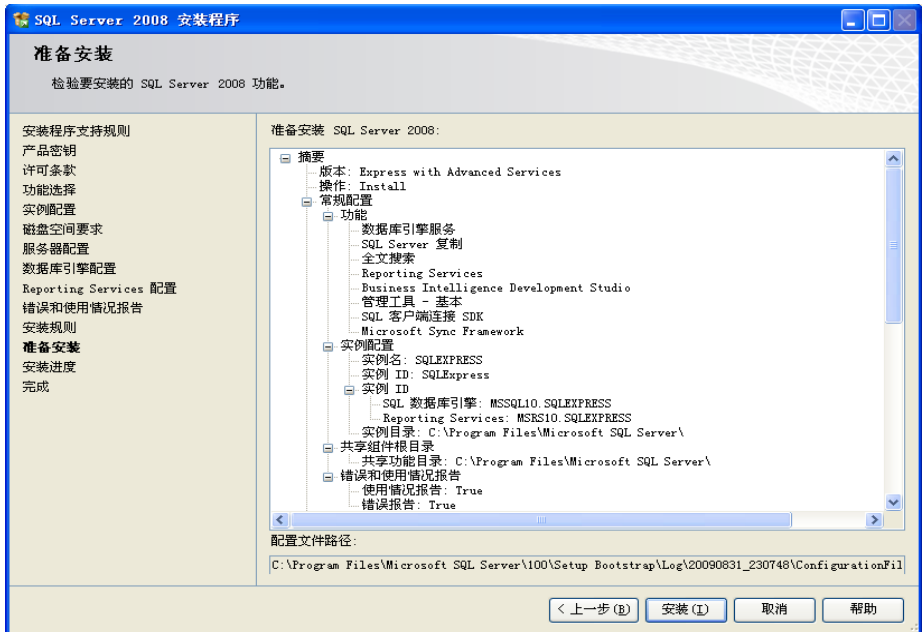


图 1-26　准备安装

（20）待确认组件列表无误后单击【安装】按钮开始安装，安装程序会根据用户对组件的选择复制相应的文件到计算机，并显示正在安装的功能名称、安装状态和安装结果，如图 1-27 所示。

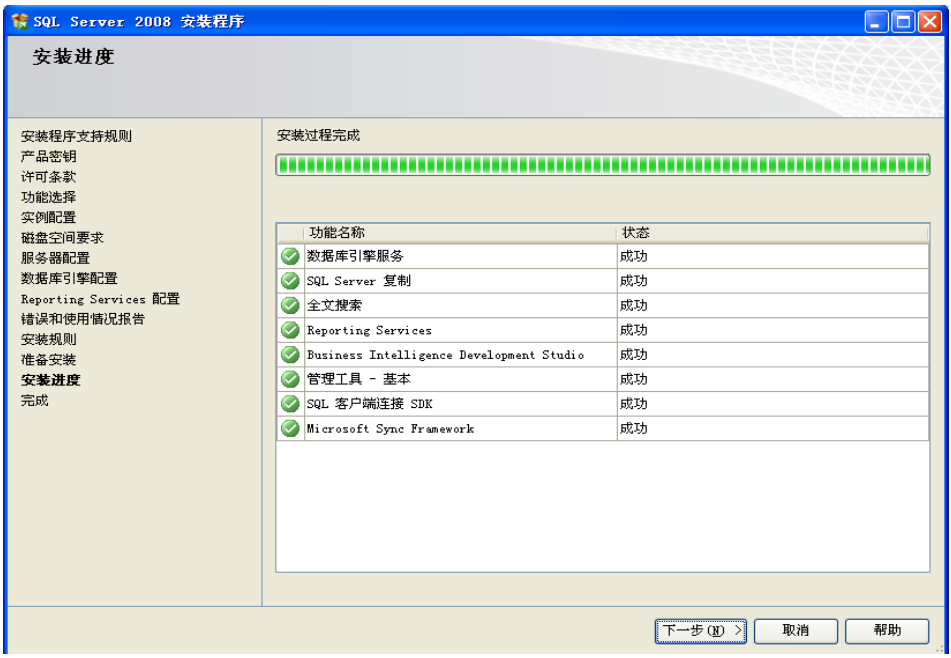


图 1-27　安装进度

（21）在图 1-27 所示的【功能名称】列表中的所有项安装成功后，单击【下一步】按钮来完成安装。此时会显示整个 SQL Server 2008 安装过程的摘要、日志保存位置及其他说明信息，

如图 1-28 所示。

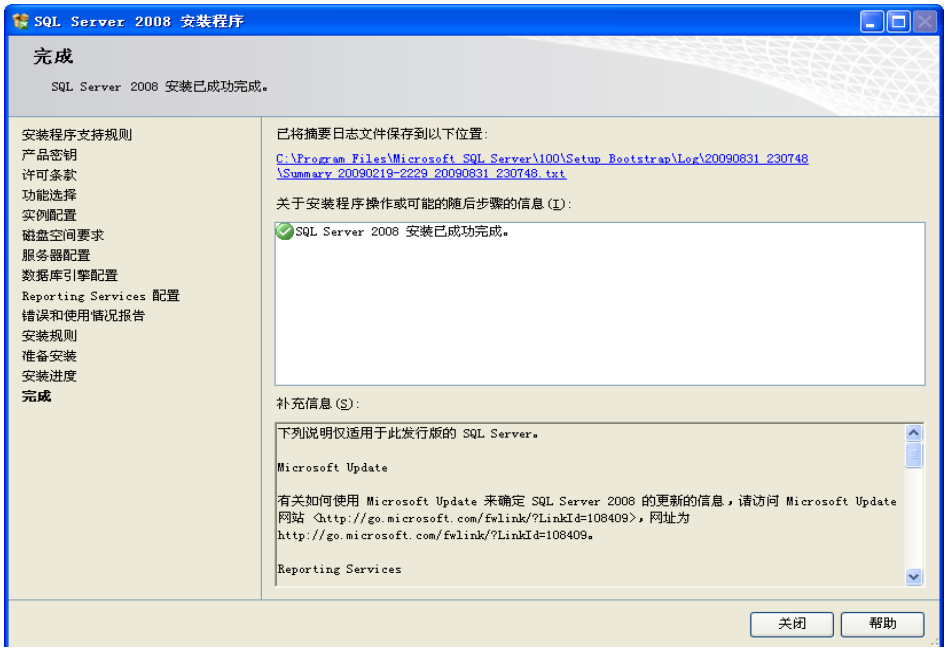


图 1-28　完成

最后，单击【关闭】按钮结束安装过程。

1.4　配置 SQL Server 2008

在 1.3 节中介绍了 SQL Server 2008 安装的过程，而安装之后的第一件事就是对 SQL Server 2008 是否安装成功进行验证，以及注册并配置 SQL Server 2008 服务器。

1．验证安装

通常情况下，如果安装过程中没有出现错误提示，即可以认为这次安装是成功的。但是，为了检验安装是否正确，也可以采用一些验证方法。例如，可以检查 Microsoft SQL Server 的服务和工具是否存在，应该自动生成的系统数据库和样本数据库是否存在，以及有关文件和目录是否正确等。

安装之后，从【开始】菜单中选择【所有程序】→【Microsoft SQL Server 2008】，可以看到如图 1-29 所示的程序组。

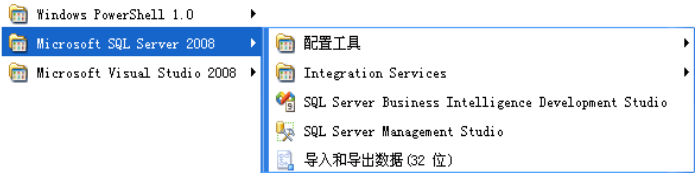


图 1-29　SQL Server 2008 程序组

SQL Server 2008 还包含了多个服务，可以通过在图 1-29 所示的菜单中单击【配置工具】→【SQL Server】命令，从弹出窗口的左侧单击【SQL Server 服务】选项来查看 SQL Server 2008

的各种服务，如图 1-30 所示。

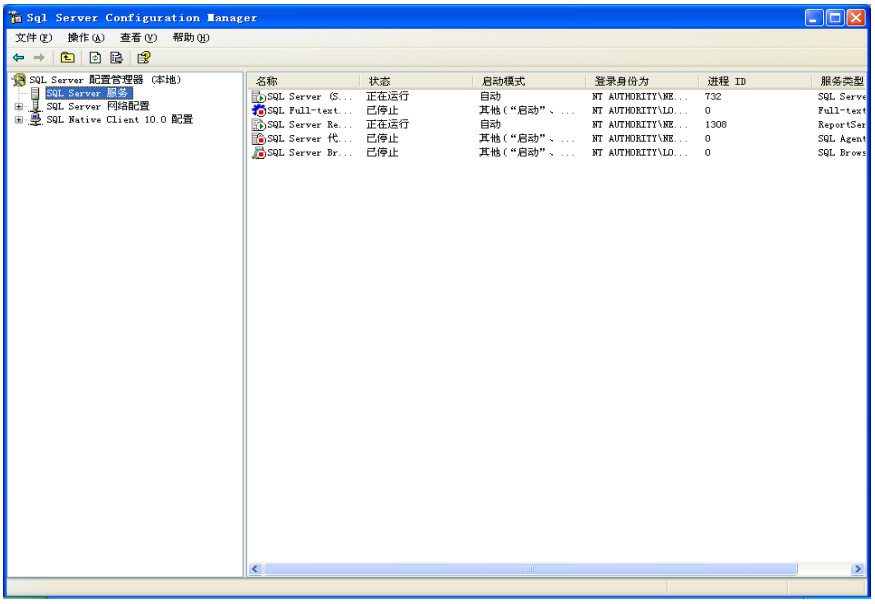


图 1-30　SQL Server 2008 的服务

2．注册服务器

注册服务器就是为 Microsoft SQL Server 客户机/服务器系统确定一台数据库所在的机器，该机器作为服务器，可以为客户端的各种请求提供服务。

（1）从【开始】菜单中选择【所有程序】→【Microsoft SQL Server 2008】→【SQL Server Management Studio】命令，打开【SQL Server Management Studio】窗口，并单击【取消】按钮。

（2）在【视图】→【已注册的服务器】窗口中展开【数据库引擎】节点，选择【Local Server Grops】→【新建服务器注册】命令，如图 1-31 所示。

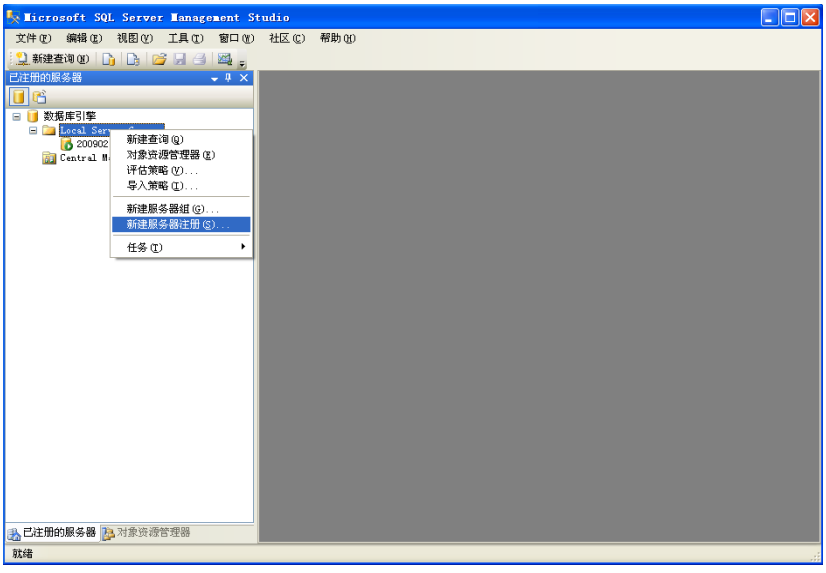


图 1-31　选择【新建服务器注册】命令

（3）打开如图 1-32 所示的【新建服务器注册】对话框，在该对话框中输入或选择要注册的服务器名称，在【身份验证】下拉列表中选择【Windows 身份验证】选项。在如图 1-33 所示的【连接属性】选项卡中可以设置链接到的数据库、网络及其他连接属性。

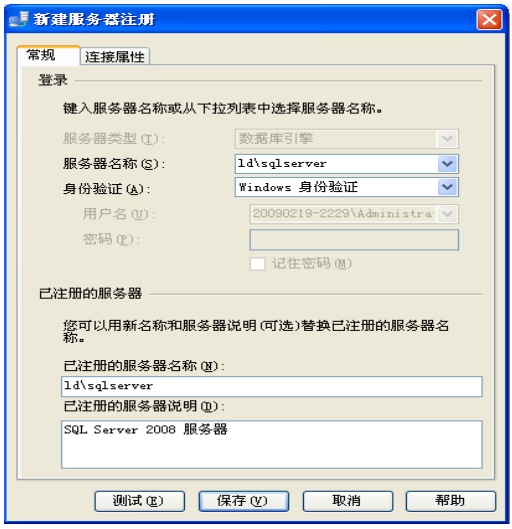


图 1-32　【新建服务器注册】对话框

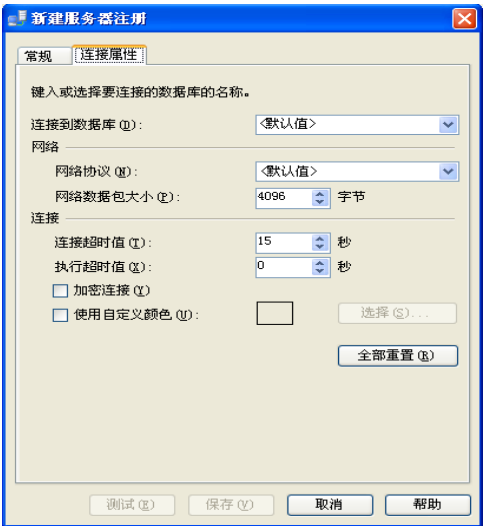


图 1-33　【连接属性】选项卡

（4）在【连接到数据库】下拉列表中指定当前用户将要链接到的数据库名称。其中，【默认值】选项表示可以从当前位置链接到 Microsoft SQL Server 系统中当前用户默认使用的数据库。【浏览服务器】表示从当前服务器中选择一个数据库。当选择【浏览服务器】选项时，打开【查找服务器上的数据库】对话框，从该对话框中可以指定当前用户连接服务器时默认的数据库。

（5）设定完成后，单击【确定】按钮返回【连接属性】选项卡，单击【测试】按钮可以验证连接是否成功，如果成功会弹出提示对话框表示连接属性的设置是正确的。

（6）最后，单击【确定】按钮返回【连接属性】选项卡，单击【保存】按钮完成注册服务器操作。

3．配置服务器

配置服务器主要是针对安装后的 SQL Server 2008 实例进行的。在 SQL Server 2008 系统中，可以使用 SQL Server Management Studio、sp_configure 系统存储过程、SET 语句等方式设置服务器选项。其中使用 SQL Server Management Studio 在图形界面中配置是最简单也是最常用的，下面以这种方法为例来进行介绍。

（1）从【开始】菜单中选择【所有程序】→【Microsoft SQL Server 2008】→【SQL Server Management Studio】命令，打开【连接到服务器】对话框，如图 1-34 所示。

（2）在此对话框的【服务器名称】文本框中输入本地计算机名称，设置【服务器类型】为【数据库引擎】，选择使用 SQL Server 或 Windows 身份验证，并验证登录名和密码。

（3）选择完成后，单击图 1-34 中的【连接】按钮，服务器将进入【Microsoft SQL Server Management Studio】窗口，表示连接成功，如图 1-35 所示。

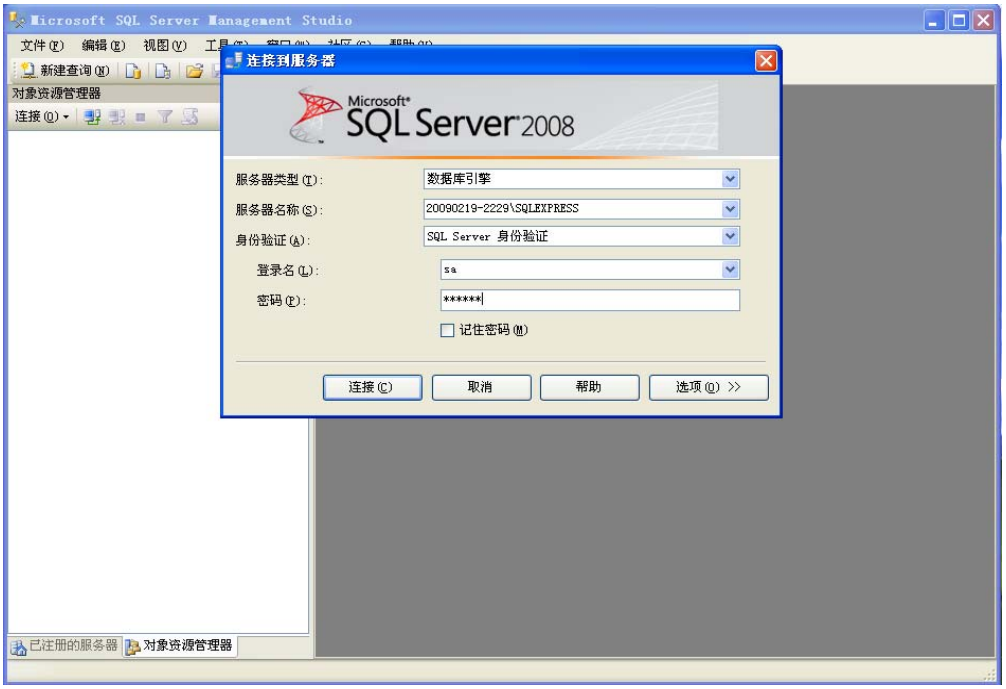


图 1-34　【连接到服务器】对话框

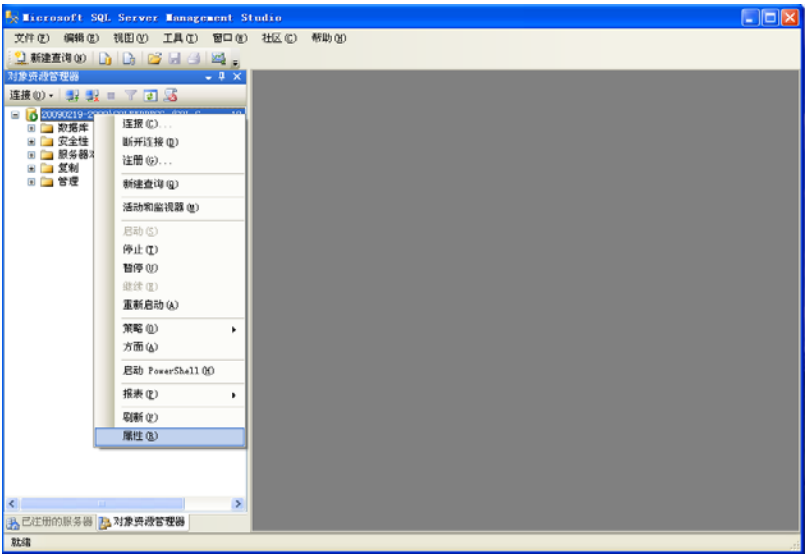


图 1-35　【Microsoft SQL Server Management Studio】窗口

（4）连接服务器成功后，右击（用鼠标右键单击）【对象资源管理器】中要设置的服务器名称，在弹出菜单中选择【属性】命令。从打开的【服务器属性】对话框中可以看出共包括了 8 个选项。其中【常规】选项窗口列出了当前服务器的产品名称、操作系统名称、平台名称、版本号、使用的语言、当前服务器的内存大小、处理器数量、SQL Server 安装的目录、服务器的排序规则，以及是否群集化等信息，如图 1-36 所示。

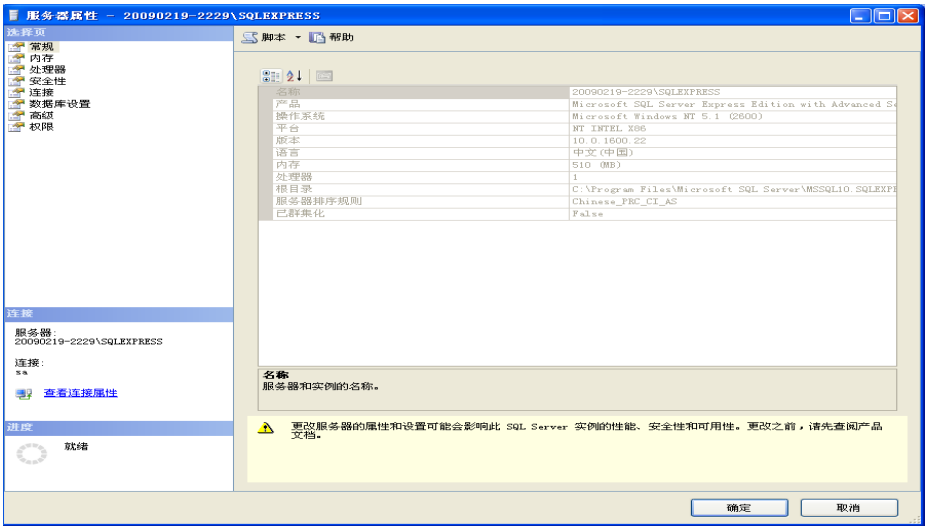


图 1-36　【服务器属性】对话框

1.5　SQL Server 2008 管理工具

在安装了 SQL Server 2008 并配置好服务器之后便可以使用了。本节将介绍随安装程序一起安装的附带管理工具和程序，它们有些是新增的，有些增强了功能。

1．SQL Server Management Studio

SQL Server Management Studio 是一个集成环境，用于访问、配置、管理和开发 SQL Server 的组件。Management Studio 使各种技术水平的开发人员和管理员都能使用 SQL Server。Management Studio 的安装需要 Internet Explorer 6.0 SP1 或更高版本。

2．Business Intelligence Development Studio

Business Intelligence Development Studio 是 Analysis Services、Reporting Services 和 Integration Services 解决方案的 IDE。Business Intelligence Development Studio 的安装需要 Internet Explorer 6.0 SP1 或更高版本。

3．SQL Server 配置管理器

SQL Server 配置管理器为 SQL Server 服务、服务器协议、客户端协议和客户端别名提供基本配置管理。

4．SQL Server Profiler

SQL Server Profiler 提供了一个图形用户界面，用于监视数据库引擎实例或 Analysis Services 实例。

5．数据库引擎优化顾问

数据库引擎优化顾问可以协助创建索引、索引视图和分区的最佳组合。

6．Reporting Services 配置管理器

Reporting Services 包括用于创建、管理和部署表格报表、矩阵报表、图形报表及自由格式报表的服务器和客户端组件。Reporting Services 还是一个可用于开发报表应用程序的可扩展平台。

7．命令提示实用工具

除了上述的图形化管理工具外，SQL Server 2008 还提供了大量的命令行实用工具，包括 bcp、dtexec、dtutil、osql、rscongif、sql、sqlwb 和 tablediff 等，下面进行简要说明。

bcp 实用工具可以在 Microsoft SQL Server 实例和用户指定格式的数据文件间大容量复制数据。使用 bcp 实用工具可以将大量新行导入 SQL Server 表，或将表数据导入数据文件。除非与 queryout 选项一起使用，否则使用该实用工具不需要了解 Transact-SQL 知识。若要将数据导入表中，必须使用为该表创建的格式文件，或者必须了解表的结构及对该表中的列有效的数据类型。

dtexec 命令提示实用工具用于配置和执行 SQL Server Integration Services 包。使用 dtexec 实用工具，可以访问所有包配置和执行功能，如连接、属性、变量、日志和进度指示器等。使用 dtexec 实用工具，可以加载来自以下 3 个源的包：Microsoft SQL Server 数据库、SSIS 服务和文件系统。

dtutil 命令提示实用工具用于管理 SQL Server Integration Services 包。该实用工具可以复制、移动和删除包，也可以验证包是否存在。它可对存储于以下 3 个位置之一的任何 SSIS 包执行上述操作：Microsoft SQL Server 数据库、SSIS 包存储区和文件系统。包的存储类型由/SQL、/FILE 和/DTS 选项标示。

osql 实用工具可以输入 Transact-SQL 语句、系统过程和脚本文件。此实用工具通过 ODBC 与服务器通信。

Rsconfig 实用工具用于配置报表服务器连接。它可以在 RSReportServer.config 文件中加密并存储连接和账户值。加密值包括用于无人参与报表处理的报表服务器数据库连接信息和账户值。

Sqlcmd 实用工具可以在命令提示符下输入 Transact-SQL 语句、系统过程和脚本文件。此实用工具使用 OLE DB 执行 Transact-SQL 批处理。在 SQL Server 2008 中通常使用 sqlcmd 来代替 osql。

Tablediff 实用工具用于比较两个表中的数据是否一致，这对于排除复制中出现的故障十分有用，也可以在批处理文件中使用该实用工具执行比较任务。

1.6　SQL Server 配置管理器

作为管理工具的 SQL Server 配置管理器（简称为配置管理器）统一包含了 SQL Server 2008 服务、SQL Server 2008 网络配置和 SQL Native Client 配置 3 个工具。它们可供数据库管理人员进行服务的启动、停止与监控，并配置服务器端支持的网络协议，还可供用户访问 SQL Server 网络，并进行相关设置。

可以通过选择【SQL Server 配置管理器】命令打开 SQL Server 配置管理器，或者通过在命令提示符下输入 sqlservermanager.msc 命令来打开 SQL Server 配置管理器。

1．配置服务

首先打开 SQL Server 配置管理器，查看列出的与 SQL Server 2008 相关的服务，选择一个并右击，选择【属性】命令进行配置。如图 1-37 所示为右击 SQL Server 后打开的【属性】对话框，在【登录】选项卡中设置服务的登录身份，并选择使用本地系统账户还是指定的账户。

切换到【服务】选项卡可以设置 SQL Server 服务的启动模式，可用选项有【自动】、【手动】和【禁用】，用户可以根据需要进行更改。

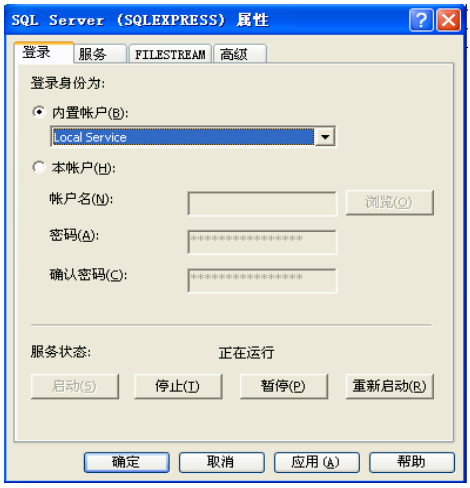


图 1-37　【属性】对话框

2．网络配置

SQL Server 2008 能使用多种协议，包括 Shared Memory、Named Pipes、TCP/IP 和 VIA，所有这些协议都有独立的服务器和客户端配置。通过 SQL Server 网络配置可以为每一个服务器实例独立地设置网络配置。

可在图 1-30 中选择左侧的【SQL Server 网络配置】节点来配置 SQL Server 服务器中所使用的协议。方法是右击一个协议名称，选择【属性】命令，在弹出的对话框中设置启用或者禁用，如图 1-38 所示为设置 Shared Memory 协议的对话框。各协议名称的含义如下。

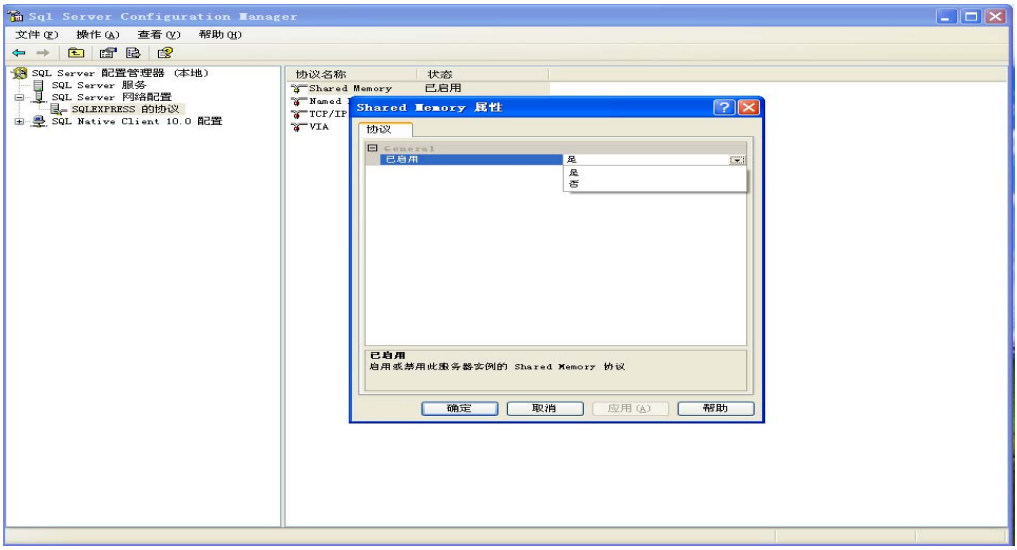


图 1-38　设置 Shared Memory 协议

1）Shared Memory 协议

Shared Memory 协议仅用于本地连接，如果该协议被启用，任何本地客户都可以使用此协议连接服务器。如果不希望本地客户使用 Shared Memory 协议，则可以禁用它。

2）Named Pipes 协议

Named Pipes 协议主要用于 Windows 2000 以前版本操作系统的本地连接及远程连接。启用 Named Pipes 时，SQL Server 2008 会使用 Named Pipes 网络库通过一个标准的网络地址作为通信，默认的实例是“\\.\pipe\sql\query”，命名实例是“\\.\pipe\MSSSQL$instancename\ sql\query”。另外，如果启用或禁用 Named Pipes，可以通过配置最高协议的属性来改变命名管道的使用。

3）TCP/IP 协议

TCP/IP 协议是通过本地或远程连接到 SQL Server 的首选协议。使用 TCP/IP 协议时，SQL Serve 会在制定的 TCP 端口和 IP 地址中侦听以响应它的请求。在默认的情况下，SQL Serve 会在所有的 IP 地址中侦听 TCP 端口 1433。每个在服务上的 IP 地址都能被立即配置，或者可以在所有的 IP 地址中监听。

4）VIA 协议

如果同一计算机上安装有两个或多个 Microsoft SQL Server 实例，VIA 连接可能会不明确。VIA 协议启用后，将尝试使用 TCP/IP 设置，并侦听端口 1433。对于不允许配置端口的 VIA 驱动程序，两个 SQL Server 实例将侦听同一端口。传入的客户端连接可能是到正确服务器实例的连接，也可能是到不正确服务器实例的连接，还有可能由于端口正在使用而被拒绝连接。因此，建议用户将该协议禁用。

3．本地客户端协议配置

通过 SQL Native Client（本地客户端协议）配置可以启用或禁用客户端应用程序使用的协议。查看客户端协议配置情况的方法是在图 1-39 所示的窗口中展开【SQL Native Client10.0 配置】→【客户端协议】节点，在右侧的信息窗口中显示了协议的名称，以及客户端连接到服务器时尝试使用的协议的顺序，如图 1-40 所示。用户还可以查看协议是否已启用或已禁用（即状态），并获得有关协议文件的详细信息。

如图 1-39 所示，在默认的情况下 Shared Memory 协议总是首选的本地连接协议。要改变协议顺序可右击一个协议，选择【顺序】命令，在弹出的【客户端协议属性】对话框中进行配置，如图 1-40 所示。从【启用的协议】列表中单击选择一个协议，然后通过右侧的两个按钮来调整协议向上或向下移动。

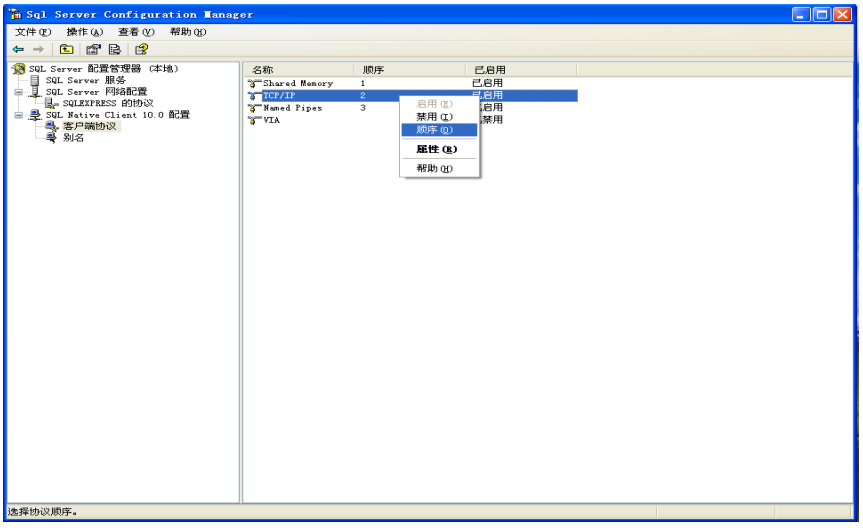


图 1-39　查看本地客户端协议

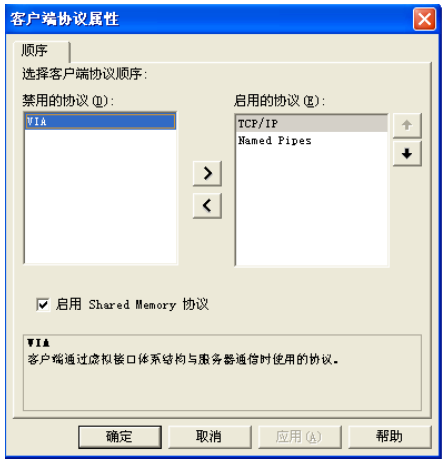


图 1-40　【客户端协议属性】对话框

1.7　SQL Server 2008 系统数据库

系统数据库就是指随安装程序一起安装，用于协助 SQL Server 系统共同管理操作的数据库，它是 SQL Server 运行的基础。在 SQL Server 2008 中，默认有 4 个系统数据库：master、model、msdb 和 tempdb。当首次安装完 SQL Server 2008 后，系统数据库就已经安装并显示出来了。

1．master 数据库

master 数据库记录 SQL Server 系统的所有系统级信息。这包括实例范围的元数据（如登录账户）、端点、链接服务器和系统配置设置。此外，master 数据库还记录了所有其他数据库的存在、数据库文件的位置及 SQL Server 的初始化信息。因此，如果 master 数据库被损坏，SQL Server 就无法启动了。在 SQL Server 2005 以后的版本中，系统对象不再存储在 master 数据库中，而是存储在 resource 数据库中。

在使用 master 数据库时，请考虑下列建议。

（1）始终有一个 master 数据库的当前备份可用。

（2）执行下列操作后，尽快备份 master 数据库：

- 创建、修改或删除任意数据库；
- 更改服务器或数据库的配置值；
- 修改或添加登录账户；
- 不要在 master 中创建用户对象，否则必须更频繁地备份 master；
- 不要针对 master 数据库将 TRUSTWORTHY 选项设置为 ON。

2．model 数据库

model 数据库用做在 SQL Server 实例上创建的所有数据库的模板。因为在创建数据库时，总是会以一套预定义的标准为模型。比如，若希望所有的数据库都有确定的初始大小，或者都有特定的信息集，那么可以把这些信息放在 model 数据库中，以 model 数据库作为其他数据库的模板数据库。如果想要使所有的数据库都有一个特定的表，也可以把该表放在 model 数据库里。由于对 model 数据库的任何改动都将反映在 tempdb 数据库中，每次启动 SQL Server

时都会创建 tempdb，所以 model 数据库必须始终存在于 SQL Server 系统中。

3．msdb 数据库

msdb 数据库由 SQL Server 代理用于计划警报和作业，也可以由其他功能（如 Service Broker 和数据库邮件）使用。

不能在 msdb 数据库中执行下列操作：

- 更改排序规则，默认排序规则为服务器排序规则；
- 删除数据库；
- 从数据库中删除 guest 用户；
- 参与数据库镜像；
- 删除主文件组、主数据文件或日志文件；
- 重命名数据库或主文件组；
- 将数据库设置为 OFFLINE；
- 将主文件组设置为 READ_ONLY。

4．tempdb 数据库

tempdb 系统数据库是一个临时性的数据库，也是一个全局资源，可供连接到 SQL Server 实例的所有用户使用，并可用于保存显式创建的临时用户对象，如全局或局部临时表、临时存储过程、表变量或游标等。它存在于 SQL Server 会话期间，一旦 SQL Server 关闭，tempdb 数据库将丢失。当 SQL Server 重新启动时，将重建全新的、空的 tempdb 数据库。

1.8　Transact-SQL 语言简介

SQL 全称是“结构化查询语言（Structured Query Language）”。SQL 是一种数据库查询和程序设计语言，用于存取数据，以及查询、更新和管理关系数据库系统。

SQL 语言是所有关系数据库通用的标准语言，而不同的数据库供应商一般都会对 SQL 语言进行不同程度的扩展，主要是基于两方面的原因：一是数据库供应商开发的系统早于 SQL 语言标准的制定时间；二是不同的数据库供应商为了达到特殊性能和实现新的功能而对标准的 SQL 语言进行了扩展。

Transact-SQL 是 SQL Server 提供的查询语言。使用 Transact-SQL 语言编写应用程序可以完成所有的数据库管理工作。任何应用程序，只要目的是向 SQL Server 的数据库管理系统发出命令以获得数据库管理系统的响应，最终都必须体现为以 Transact-SQL 语句为表现形式的指令。对用户来说，Transact-SQL 是唯一可以和 SQL Server 的数据库管理系统进行交互的语言。

尽管 SQL Server 2008 提供了使用方便的图形化用户界面，但各种功能的实现基础还是 Transact-SQL 语言，只有 Transact-SQL 语言可以直接和数据库引擎进行交互。Transact-SQL 语言是基于商业应用的结构化查询语言，是标准 SQL 语言的增强版本。

习　　题

1．解释下列概念：数据、数据库、数据库管理系统、数据库系统。

2．简述数据库系统的特点。

3．简述概念模型的作用，并解释以下术语：实体、实体性、实体集、码、属性、联系。

4．为某连锁商店设计一个 E-R 模型。该企业下设有若干连锁店，每家商店有若干职工，但每个职工只能服务于一家商店。其中商店实体型的属性包括商店号、商店名、地址、店长。商品实体型的属性包括商品号、商品名、规格、单价、产地。职工实体型的属性包括职工号、姓名、性别、工资。在联系中应反映出职工参加某商店的开始时间、商店销售商品的月销售量。

5．简述 SQL Server 2008 安装过程。

6．列举 SQL Server 2008 管理工具，并描述其功能。

第 2 章　数据库和表创建

- 认识数据库和表
- 了解数据类型
- 掌握数据库和数据表的创建

2.1　SQL Server 基本概念

2.1.1　数据库

SQL Server 是典型的关系数据库，它由表、视图、索引、存储过程、用户和触发器等对象组成。SQL Server 数据库分为 4 类，即系统数据库、Report Server 数据库、Report Server 临时数据库和用户数据库。系统数据库和示例数据库是在安装了 SQL Server 后自动创建的数据库。用户数据库是用户创建的数据库，其结构与系统数据库相同。

1．系统数据库

系统数据库存储了 SQL Server 的系统信息，SQL Server 的系统数据库有 master、model、msdb 和 tempdb。

（1）master 数据库：是 SQL Server 的主数据库，它存储了 SQL Server 的系统级信息，包括登录号、系统配置、数据库位置及数据库错误信息等，用于控制用户数据库和 SQL Server 的运行。

（2）msdb 数据库：为 SQL Server 代理调度信息，并为作业记录提供存储空间。

（3）model 数据库：为新创建的数据库提供模板。

（4）tempdb 数据库：是临时数据库，提供了临时存储空间，所有连接到系统的用户临时表和临时存储过程都存储在此，存储的这些对象不会永远保存。SQL Server 关闭时，tempdb 数据库中的所有对象会自动被删除。

2．Report Server 数据库

Report Server 数据库是一个 SQL Server 数据库，它能够存储 SSRS 配置部分、报告定义、报告元数据、报告历史、缓存政策、快照、资源、安全设置、加密的数据、调度和提交数据，以及扩展信息。

3．Report Server 临时数据库

Report Server 临时数据库负责存储中间处理产品，如缓冲的报告、会话和执行数据等。

2.1.2　表

1．认识数据表

“什么是数据表？它和数据库之间是什么关系呢？”此时，大家肯定都有这样的疑问，下面就来解答。数据库是由各种数据表组成的，数据表是 SQL Server 数据库中最重要的对象，用来存储和操作数据的逻辑结构。表由列和行组成，列是表数据的描述，行是表数据的实例。

如图 2-1 所示是某个数据库的信息表，可以看出表由行和列组成，因此也称为二维表。行

对应的是记录，列对应的是字段。每个表中都有一个用于识别不同记录的字段，这个字段叫做关键字。

	Cno	Cname	Ccredit	Cteacher	Chours
1	1	DB_Design	2	NULL	36
2	2	VB_Programming	2	NULL	36
3	3	MIS	4	NULL	72
4	4	BIT	3	NULL	54

图 2-1　数据库信息表

- 记录：表包含若干行数据，表中的一行数据称为一个记录，因此表也是记录的集合。
- 字段：字段也称为域，数据表中的每个记录由多个字段组成，同一个表中的不同记录应包含相同的字段，每个字段的值可能包含不同类型的数据。
- 关键字：关键字是用于标记该表中记录唯一性的字段，即其值是不允许重复的，如人的身份证号一样，每个表必须具有一个关键字。

2．数据类型

在 SQL Server 2008 中，每个列、局部变量、表达式和参数都具有一个相关的数据类型。数据类型是一种属性，用于指定对象可保存的数据的类型，如整数数据、字符数据、货币数据、日期和时间数据、二进制字符串等。SQL Server 2008 中的系统数据类型如表 2-1 所示。

表 2-1　数据类型

数据类型	包括的类型
精确数字	numeric，smallint，smallmoney，tinyint，bigint，bit，decimal，int，money
近似数字	float，real
日期和时间	date，datetime2，datetime，datetimeoffset，smalldatetime，time
字符串	char，varchar，text
Unicode 字符串	nchar，nvarchar，ntext
二进制字符串	binary，varbinary，image
其他数据类型	cursor，hierarchyid，sql_variant，table，timestamp，uniqueidentifier，xml

1）精确数字型

（1）整数型：用于存储精确的整型数据，包括负整数、零和正整数。整数型包括 int、bigint、smallint 和 tinyint 类型。

- bigint：-2^{63} 到 $2^{63}-1$ 之间的有符号整数，长度 8 字节。
- int：-2^{31} 到 $2^{31}-1$ 之间的有符号整数，长度 4 字节。
- smallint：-2^{15} 到 $2^{15}-1$ 之间的有符号整数，长度 2 字节。
- tinyint：0 到 255 之间的无符号整数，长度 1 字节。

（2）货币型：用于存储货币类型数据，如工资、价格等。货币型数据类型包括 money 和 smallmoney。

- money：-2^{63} 到 $2^{63}-1$ 之间的数，长度 8 字节。
- smallmoney：-2^{31} 到 $2^{31}-1$ 之间的数，长度 4 字节。

（3）精确数值型：精确数值型是带固定精度和小数位数的数值数据类型，包括

decimal[(p[,s])]和 numeric[(p[,s])]两类。

其中 p 表示精度，即最多可以存储的十进制数字的总位数，包括小数点左边和右边的位数。该精度必须是从 1 到最大精度 38 之间的值，默认精度为 18。s 表示小数位数，即小数点右边可以存储的十进制数字的最大位数。小数位数必须是从 0 到 p 之间的值。仅在指定精度后才可以指定小数位数。默认的小数位数为 0，因此 0≤s≤p。最大存储大小基于精度而变化。

- 精度为 1～9 时，存储长度为 5 字节。
- 精度为 10～19 时，存储长度为 9 字节。
- 精度为 20～28 时，存储长度为 13 字节。
- 精度为 29～38 时，存储长度为 17 字节。

例如，decimal(3,1)表示实际存储长度为 5 字节，decimal(11,5)表示实际存储长度为 9 字节。

（4）位型：SQL Server 2008 中的位型数据相当于其他语言中的逻辑型数据，包括 1、0 和 NULL。

例如，表中有一列为性别，性别只有两种可能性，这时可以把性别这列的数据类型设为 bit。

2）近似数字数值型

用于表示浮点数值数据的大致数值的数据类型，包括 float 和 real 两类。

（1）float：−1.79E+308 至−2.23E−308、0，以及 2.23E−308 至 1.79E+308 之间的浮点数，长度取决于 n 值。

（2）real：−3.40E+38 至−1.18E−38、0，以及 1.18E−38 至 3.40E+38 之间的浮点数，长度为 4 字节。

3）日期和时间型

当需要存储日期或时间信息时，需要使用日期和时间型数据类型，包括 date、datetime2、datetime、datetimeoffset、smalldatetime 和 time。

（1）time：格式为 hh:mm:ss，范围为 00:00:00.0000000～23:59:59.9999999，精确度为 100 纳秒，存储字节为 3～5。

（2）date：格式为 YYYY.MM.DD，范围为 0001.01.01～9999.12.31，精确度为 1 天，存储字节为 3。

（3）smalldatetime：格式为 YYYY.MM.DD hh:mm:ss，范围为 1900.01.01 00:00:00～2079.06.06 23:59:59，精确度为 1 分钟，存储字节为 4。

（4）datetime：格式为 YYYY.MM.DD hh:mm:ss[.nnn]，范围为 1753.01.01 00:00:00～9999.12.31 23:59:59.997，精确度为 0.003 33 秒，存储字节为 8。

（5）datetime2：格式为 YYYY.MM.DD hh:mm:ss，范围为 0001.01.01 00:00:00.0000000～9999.12.31 23:59:59.9999999，精确度为 100 纳秒，存储字节为 6～8。

（6）datetimeoffset：格式为 YYYY.MM.DD hh:mm:ss [+|−]hh:mm，范围为 0001.01.01 00:00:00.0000000～9999.12.31 23:59:59.9999999，精确度为 100 纳秒，存储字节为 8～10。

4）字符型

字符型数据是字符串，包括英文字母、符号、汉字和数字等。SQL Server 2008 的字符类型包括 char、varchar 和 text。

（1）char(n)：用来存放固定长度的字符数据，如身份证号码、电话号码和编号等。可以用 n 来指定字符串的长度，n 的取值范围为 1～8 000。

（2）varchar(n)：用来存放可变长度的 n 个字符，如姓名、风景名胜的名字和地址等。n 表示字符串可达到的最大长度。n 的取值范围为 1～8 000。此种数据类型的数据长度为输入的字符串的实际长度，而不一定是 n。

（3）text：用于存储的字符超过 8 000 个时使用，如备注等。此类型的最大长度为 2 147 483 647 个字符。其数据的长度为实际的字符个数。

5）unicode 字符串型

unicode 字符串型数据是符合 Unicode 标准字符集定义的所有字符，它的优点是可以使计算机的代码统一，如中文、英文等任何字符都可以出现在一行中，包括 nchar、nvarchar、ntext。

（1）nchar(n)：用于存储固定长度的 unicode 数据，n 值必须在 1 到 4 000 之间（含）。存储大小为 n 字节的两倍。

（2）nvarchar(n)：用于存储可变长度的 unicode 数据，n 值在 1 到 4 000 之间（含）。max 指示最大存储大小为 $2^{31}-1$ 字节。存储大小是所输入字符个数的两倍再加两个字节。所输入数据的长度可以为 0 个字符。

（3）ntext(n)：当一个字段中存储的字符超过 4 000 个且为可变长度的 Unicode 文本数据时，应该选择 ntext 类型，其最多可以存 $2^{30}-1$（1 073 741 823）个字符，存储大小是所输入字符个数的两倍（以字节为单位）。

6）二进制字符串型

用于存储二进制数据，如声音、图片等，包括 binary、varbinary 和 image 类型。

（1）binary(n)：用于存储长度为 n 字节的固定长度二进制数据，其中 n 可以是从 1 到 8 000 之间的值。存储大小为 n 字节。

（2）varbinary[(n|max)]：用于存储可变长度二进制数据。n 可以是从 1 到 8 000 之间的值。max 指示最大存储大小为 $2^{31}-1$ 字节。存储大小为所输入数据的实际长度加两个字节。所输入数据的长度可以是 0 字节。

（3）image(n)：假如要存储的字段超过 8 000 字节且为可变长度的二进制数据，则采用 image 数据类型，如图片文件、OLE 对象等，其最大长度为 $2^{31}-1$（2 147 483 647）个字节。

2.2　用界面方式创建数据库和表

通过 2.1 节的学习，大家对数据库和表有了一定的了解，下面学习如何用界面方式创建一个数据库，以及在数据库内创建一张表。

2.2.1　数据库的创建、修改和删除

1．数据库的创建

在 SQL Server 2008 中，通过 SQL Server Management Studio 创建数据库是最容易的方法，对初学者来说简单易用。下面以创建本书的示例数据库 db_stu 为例，对这种方法进行详细介绍。具体的操作步骤如下。

（1）从【开始】菜单中选择【所有程序】→【Microsoft SQL Server 2008】→【SQL Server Management Studio】命令，打开【连接到服务器】对话框，并使用 Windows 或 SQL Server 身份验证建立连接，如图 2-2 所示。

图 2-2　连接服务器身份验证

（2）在【对象资源管理器】窗口中展开服务器，然后选择【数据库】节点。

（3）在【数据库】节点上单击鼠标右键，从弹出的快捷菜单中选择【新建数据库】命令，如图 2-3 所示。

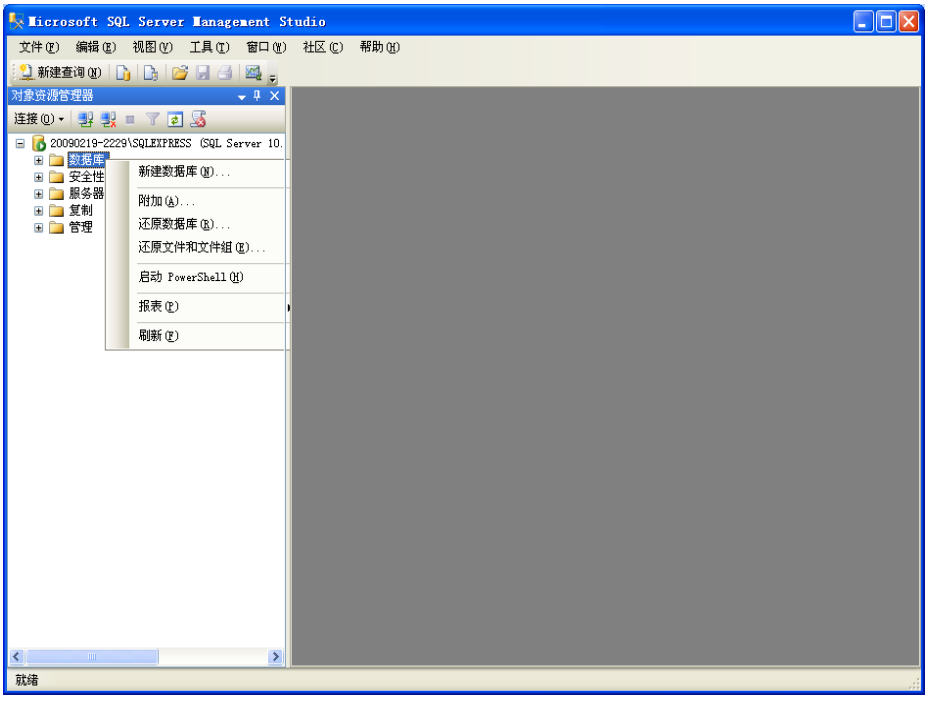


图 2-3　选择【新建数据库】命令

（4）执行上述操作后，会弹出【新建数据库】对话框，如图 2-4 所示。这个对话框有 3 个选项，分别是【常规】、【选项】和【文件组】。完成这 3 个选项中的内容之后，就完成了数据库的创建工作。

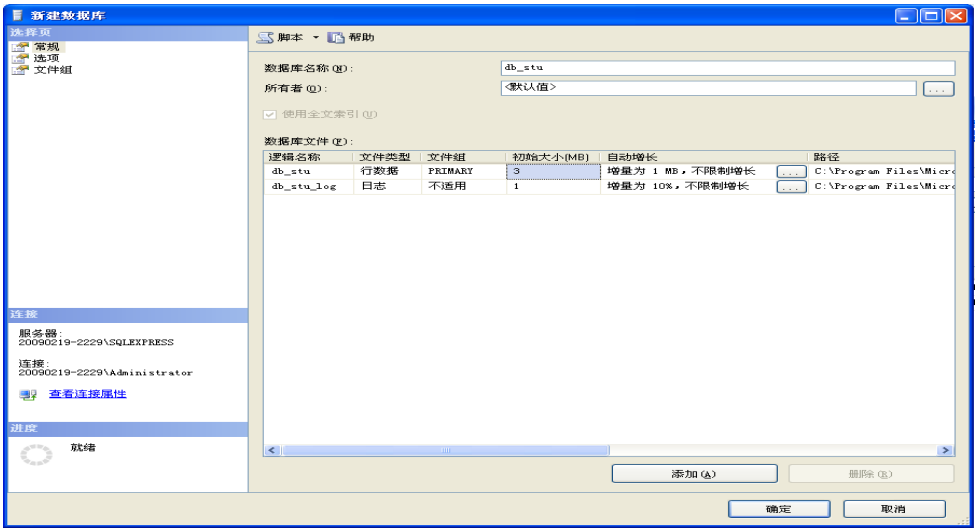


图 2-4　【新建数据库】对话框

（5）在【数据库名称】文本框中输入要新建数据库的名称。

（6）在【所有者】文本框中输入新建数据库的所有者，如 sa。根据数据库的使用情况，选择启用或者禁用【使用全文索引】复选框。

（7）【数据库文件】列表包括两行，一行是数据文件，另一行是日志文件。通过单击相应的按钮，可以添加或删除相应的数据文件。该列表中各字段值的含义如下。

- 逻辑名称：指定该文件的文件名，其中数据文件与 SQL Server 2000 不同，在默认情况下不再为用户输入的文件名添加下画线和 Data 字样，但相应的文件扩展名并未改变。
- 文件类型：用于区别当前文件是数据文件还是日志文件。
- 文件组：显示当前数据库文件所属的文件组。一个数据库文件只能存在于一个文件组中。
- 初始大小：制定该文件的初始容量，在 SQL Server 2008 中数据文件的默认大小为 3MB，日志文件的默认大小为 1MB。
- 自动增长：用于设置当容量不够用时，文件根据何种增长方式自动增长。通过单击【自动增长】列中的省略号按钮，可打开更改自动增长设置对话框进行设置。如图 2-5 和图 2-6 所示分别为数据文件和日志文件的自动增长设置对话框。

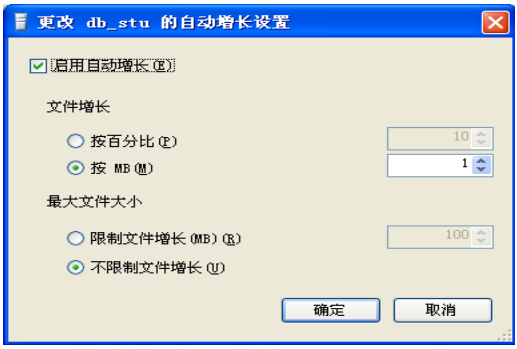


图 2-5　数据文件自动增长设置

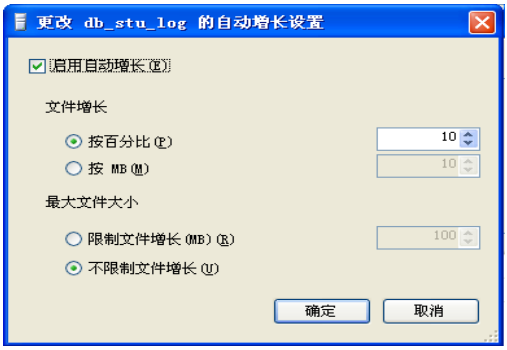


图 2-6　日志文件自动增长设置

- 路径：指定存放该文件的目录。在默认情况下，SQL Server 2008 将存放路径设置为 SQL

Server 2008 安装目录下的 data 子目录。单击该列中的按钮可以打开【定位文件夹】对话框更改数据库的存放路径。

（8）选择【选项】页，可设置数据库的排序规则、恢复模式、兼容级别和其他需要设置的内容，如图 2-7 所示。

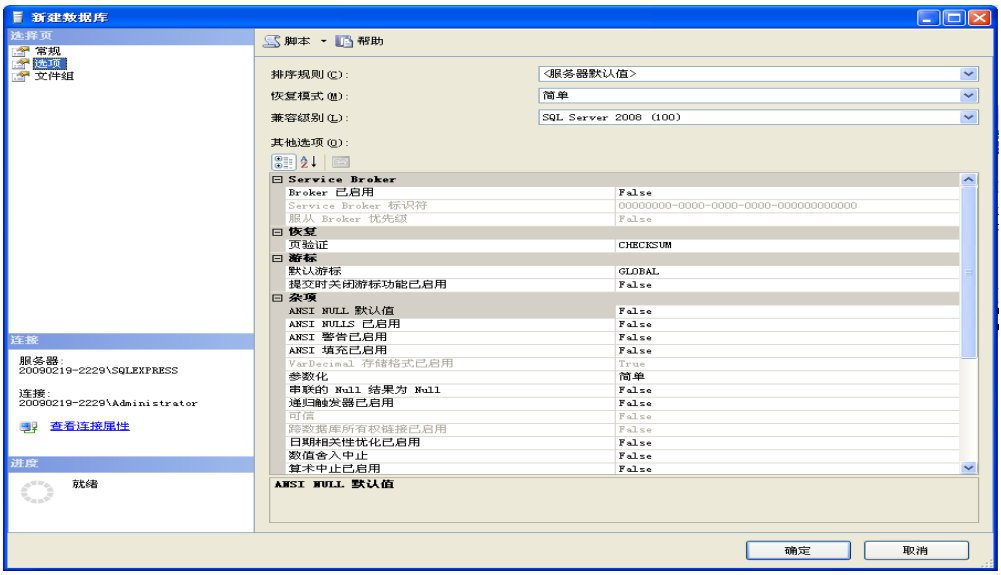


图 2-7　【选项】页

（9）单击【文件组】可以设置数据库文件所属的文件组，可以通过【添加】或者【删除】按钮更改数据库文件所属的文件组，如图 2-8 所示。

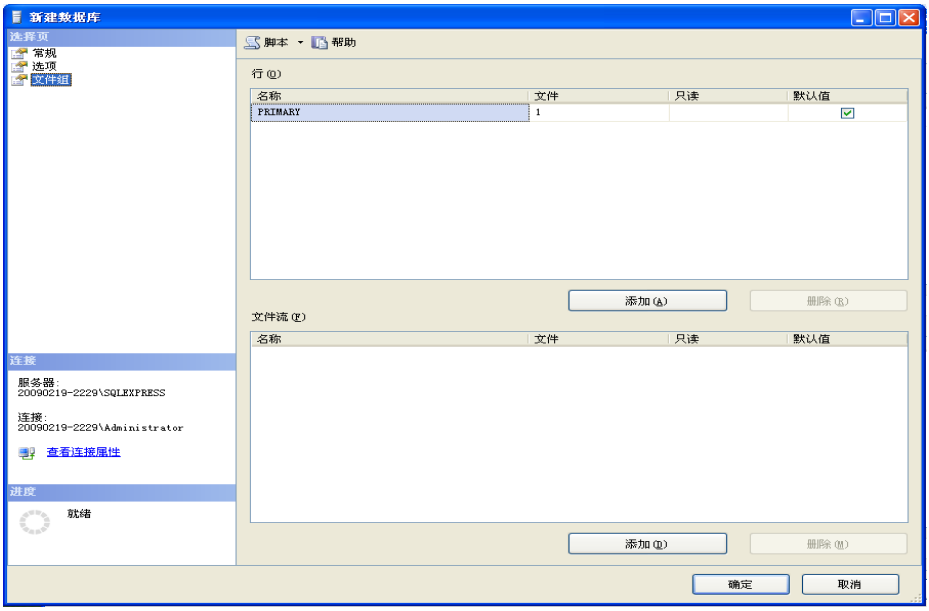


图 2-8　【文件组】页

（10）完成以上操作后，就可以单击【确定】按钮关闭【新建数据库】对话框了。这时，

一个数据库就成功地创建了，可以通过【对象资源管理器】窗口查看新建的数据库。

2．数据库的修改

1）数据库更名

通常情况下，创建好一个数据库后就不再更改其数据库名称了。因为许多应用程序可能已经使用该名称，如果要更改数据库名称，所有引用其名称的应用程序都要做相应的修改。但是，如果确实需要修改数据库名称，也有很多方法，下面介绍使用图形界面修改数据库名称的方法。

使用图形界面是修改数据库名称最简单的一种方法，只要在【对象资源管理器】窗口中右击一个要修改的数据库，选择【重命名】命令，即可直接改名。

2）利用 SQL Server 管理控制台修改数据库属性

下面介绍如何利用 SQL Server 管理控制台的图形界面修改数据库的属性。

（1）在【对象资源管理器】窗口中，右击要修改的数据库，选择【属性】命令。

（2）在【数据库属性】对话框的【选择】页下选择【文件】选项。

（3）在要修改的数据库数据文件行的【初始大小】列中，输入要修改的值。同样在日志文件行的【初始大小】列中，输入要修改的值。

（4）单击【自动增长】列中的按钮，打开【自动增长设置】对话框，可设置自动增长的方式及大小。

（5）如果要添加文件，可以直接在【数据库属性】对话框中单击【添加】按钮，进行相应的设置。

（6）完成修改后，单击【确定】按钮完成对数据库的修改。

3）缩小数据库

如果数据库的设计尺寸过大或者删除了数据库中的大量数据，数据库会浪费大量的磁盘资源。根据用户的实际需要，可以对数据库进行缩小。在 Microsoft SQL Server 2008 系统中，缩小数据库有以下 3 种方式。

（1）使用 AUTO_SHRINK 数据库选项设置自动缩小数据库。

将 AUTO_SHRINK 选项设置为 ON 后，数据库引擎将自动缩小具有可用空间的数据库。此选项可以使用 ALTER DATABASE 语句进行设置。默认情况下，此选项设置为 OFF。数据库引擎会定期检查每个数据库的空间使用情况，如果某个数据库的 AUTO_SHRINK 选项设置为 ON，数据库引擎将自动减小数据库中的文件。设置 AUTO_SHRINK 选项的语法格式如下：

```
ALTER DATABASE database_name SET AUTO_SHRINK ON
```

（2）使用 DBCC SHRINKDATABASE 命令缩小数据库。

这种方式要求手动缩小数据库的大小，它是一种比自动缩小数据库更加灵活的方式，可以对整个数据库进行缩小。DBCC SHRINKDATABASE 命令的基本语法格式如下：

```
DBCC SHRINKDATABASE('database_name',target_percent)
```

（3）使用 DBCC SHRINKDFILE 命令缩小数据库文件。

该命令用于缩小指定的数据库文件，也可以将文件缩小至小于其初始创建大小，并且重新设置当前大小为其初始创建大小。DBCC SHRINKDFILE 命令的基本语法格式如下：

```
DBCC SHRINKDFILE('file_name',target_size)
```

3．数据库的删除

删除数据库就是从 SQL Server 系统中删除不再使用的数据库。使用图形界面删除数据库的具体步骤如下。

（1）在【对象资源管理器】窗口中选中要删除的数据库，单击鼠标右键，在弹出的菜单中单击【删除】命令。

（2）在弹出的【删除对象】对话框中，单击【确定】按钮确认删除。删除操作完成后会自动返回【SQL Server Management Studio】窗口。

2.2.2　表的创建、修改和删除

1．概述

创建表就是对表中列的操作，列也称字段，因此创建表就是定义字段和关键字的过程。

1）定义字段

定义字段就是字段的命名和属性的设置，这是创建数据表最重要的步骤，字段的属性的含义如下。

（1）列名：定义字段的名称。

（2）数据类型：定义字段的数据类型，默认数据类型为 nchar。

（3）长度：定义字段可存放数据的长度，默认长度为 10，单位为“字节”。

（4）允许空：定义该字段值是否可以为 NULL，默认允许为 NULL。

（5）默认值或绑定：定义字段的默认值。

（6）说明：定义字段的说明信息。

（7）精度：显示数值数据类型所允许的最大位数。对于非数值数据类型，此属性显示为 0。

（8）小数位数：显示数值数据类型的小数点右侧可显示的最大位数。此值必须小于或等于精度。对于非数值数据类型，此属性显示为 0。

（9）标志：显示所选列是否为表的标志列。若要更改属性，请在表设计器中打开表，然后在属性窗口中编辑这些属性。此设置仅适用于基于数字的数据类型的列，如 int。

（10）标志增量：显示各后续行的“标志种子”要增加的增量。如果将此单元格保留为空白，则默认情况下，会将值 1 赋给该单元格。若要编辑此属性，请直接输入新值。

（11）标志种子：显示分配给表中第一行的值。如果将此单元格保留为空白，则默认情况下，会将值 1 赋给该单元格。若要编辑此属性，请直接输入新值。

（12）大小：显示该列的数据类型所允许的大小（字节）。例如，某个 nchar 数据类型的长度为 10（字符数），但在 Unicode 字符集中，该数据类型的大小为 20。

（13）公式：定义计算字段的表达式，在定义表达式时，字段名称应使用【】括起来，如【a】+【b】+3，其中 a 和 b 分别为字段的名称。

（14）排序规则：显示所选列的排序规则设置。若要更改此设置，请单击“排序规则”，再单击值右侧的省略号（…）。

在表设计器窗口定义表中的各个字段，如图 2-9 所示。

2）定义主关键字

定义主关键字可以定义表的唯一性，其操作如下。

（1）在表设计器窗口中单击字段左边的　　按钮，选择该字段，如图 2-10 所示。

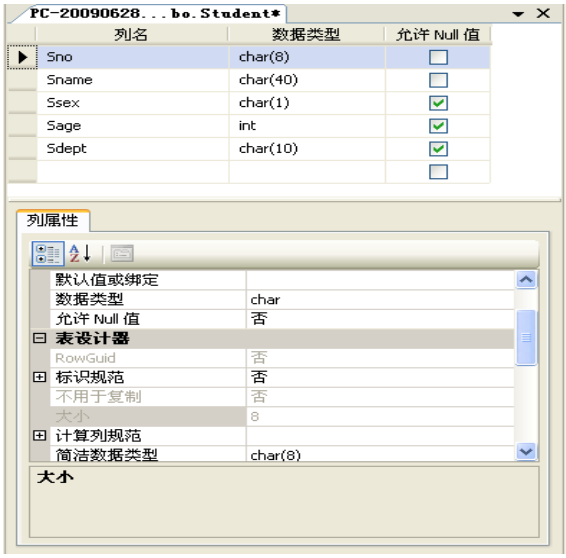


图 2-9　表设计器窗口

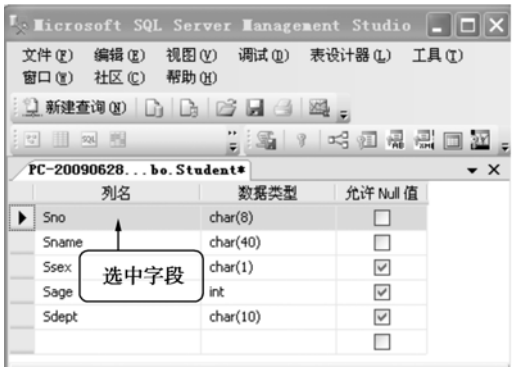


图 2-10　选中字段

（2）在工具栏中单击 按钮，将选中的字段设置为表的主键，如图 2-11 所示，或在选中的字段上的任何地方单击鼠标右键，从弹出菜单中选择【设置主键】命令，如图 2-12 所示。主键的选择按钮上会显示一个钥匙图标。

2．使用 SQL Server 管理控制台创建表

在【SQL Server Management Studio】中，可在表设计器中创建数据表，具体操作如下。

（1）在【对象资源管理器】中，右击数据库的【表】节点，再单击【新建表】命令，如图 2-13 所示。

（2）输入列名，选择数据类型，并选择各个列是否允许空值，如图 2-14 所示。

（3）在【文件】菜单中选择【保存】命令，如图 2-15 所示。

（4）在【选择名称】对话框中，为该表输入一个名称，再单击【确定】按钮，如图 2-16 所示。

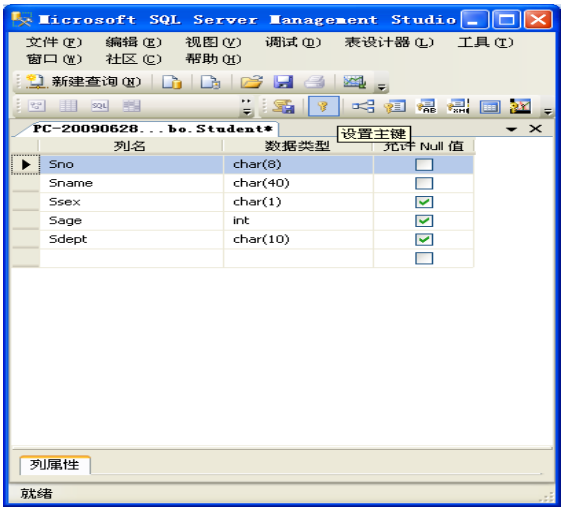


图 2-11　定义主键方法一

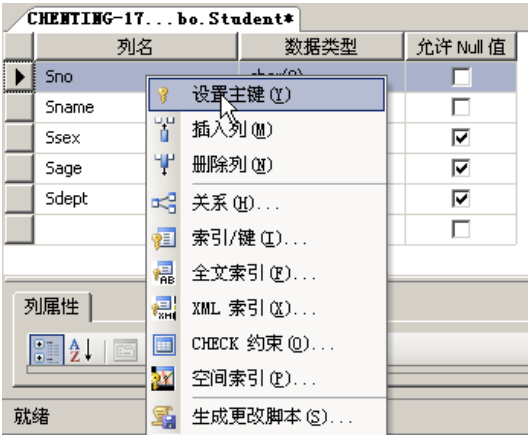


图 2-12　定义主键方法二

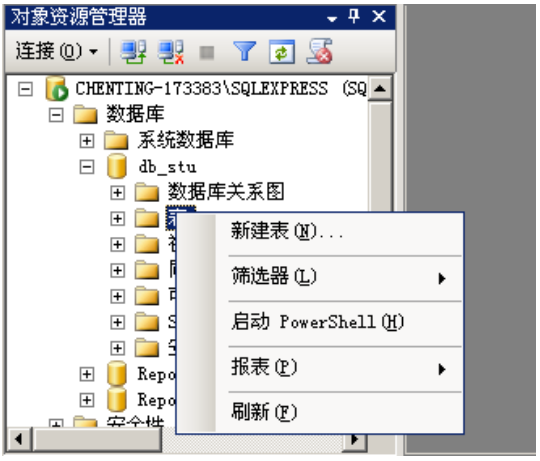


图 2-13　新建表

图 2-14　定义字段

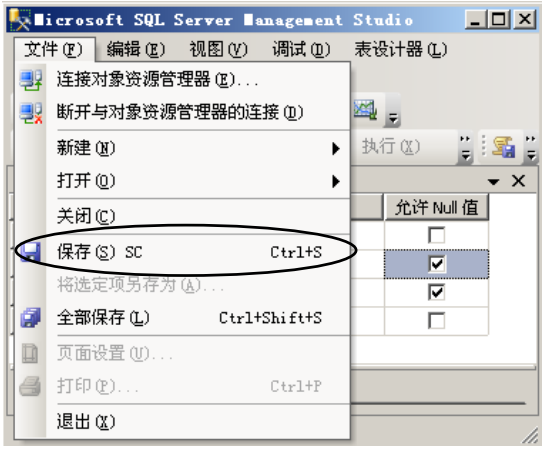


图 2-15　保存表

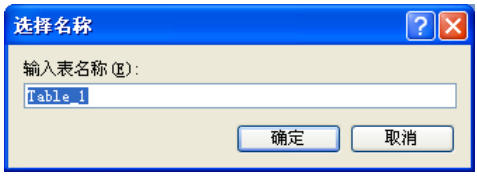


图 2-16　【选择名称】对话框

3．使用 SQL Server 管理控制台修改表

使用 SQL Server 管理控制台修改表的具体操作步骤如下。

（1）在【对象资源管理器】中，选择数据库【db_stu】，在【表】中选择【dbo.Student】，并且单击鼠标右键，在弹出的快捷菜单中选择【设计】命令，打开表【Student】，如图 2-17 所示。

（2）直接在表设计器窗口中的【Sname】字段后添加【Sage】字段，如图 2-18 所示。

（3）选择字段【Sage】，单击鼠标右键，在弹出的快捷菜单中选择【插入列】命令，插入一个空行，在插入的空行中的【列名】处输入【Ssex】，并设置其属性，如图 2-19 所示。

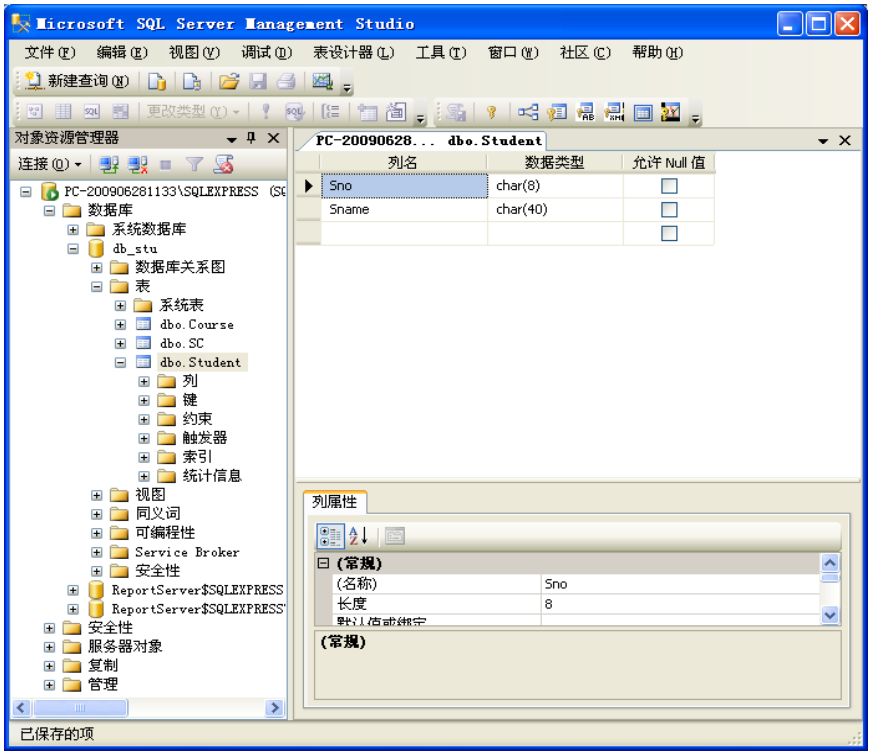


图 2-17　打开表 Student

图 2-18　添加字段

图 2-19　插入字段

（4）用同样的方法创建【Sdept】字段。

（5）选择字段【Sdept】，单击鼠标右键，在弹出的快捷菜单中选择【删除列】命令，删除【Sdept】字段，如图 2-20 所示。

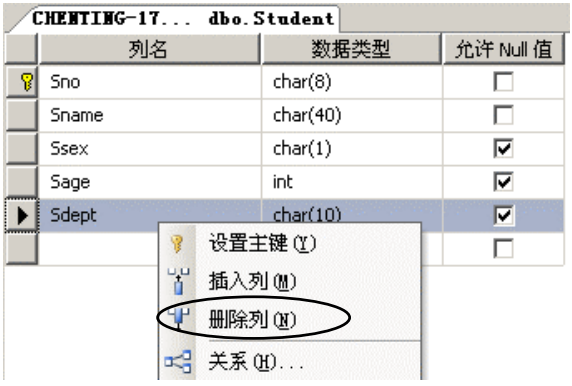


图 2-20　【删除列】命令

（6）单击字段【Sage】左边的▶按钮，按住鼠标左键不放，将其拖动到新的位置，释放鼠标，即可完成位置的移动，如图 2-21 所示。

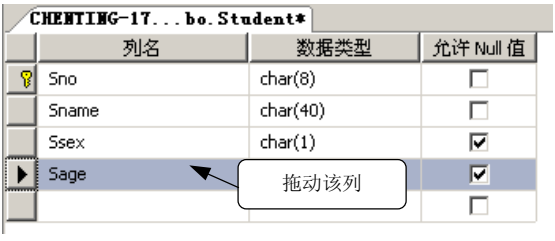


图 2-21　调整字段顺序

4．使用 SQL Server 管理控制台删除表

使用 SQL Server 管理控制台删除表的具体操作步骤如下。

（1）在【对象资源管理器】中选择要删除的表，然后按【Del】键，或选择【编辑】→【删除】命令，或单击鼠标右键，在弹出的快捷菜单中选择【删除】命令，打开【删除对象】对话框。

（2）在该对话框中选择要删除的对象，单击 确定 按钮即可，如图 2-22 所示。

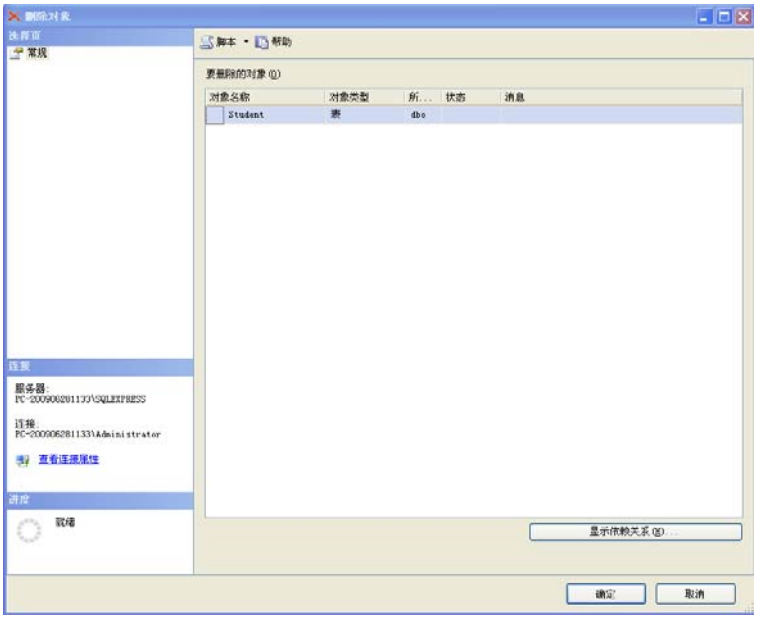


图 2-22　删除表

2.3　使用命令方式创建数据库和表

2.3.1　使用 CREATE DATABASE 创建数据库

通过两个例子来介绍如何使用 CREATE DATABASE 语句创建数据库。

【例 2.1】　创建一个名为 db_stu 的数据库，其初始大小为 5MB，最大为 50MB，允许数据库自动增长，增长方式是按 10%的比例增长；日志文件初始为 2MB，最大可增长到 5MB，按 1MB 增长。假设 SQL Server 服务已启动，并以 Administrator 身份登录计算机。

（1）打开【Microsoft SQL Server Management Studio】窗口，并连接到服务器。

（2）选择【文件】→【新建】→【数据库引擎查询】命令或者单击标准工具栏上的【新建查询】按钮，创建一个查询输入窗口。

（3）在窗口内输入语句创建数据库，保存位置为“E:\sql\data\MSSQL\Data”。

CREATE　DATABASE 语句如下：

```
CREATE　DATABASE　db_stu
ON
(
NAME='db_stuData',
```

```
FILENAME='e:\sql\data\MSSQL\Data\db_stu.mdf',
SIZE=5MB,
MAXSIZE=50MB,
FILEGROWTH=10%
),
LOG ON
(
NAME='db_stuLog',
FILENAME='e:\sql\data\MSSQL\Data\db_stu.ldf',
SIZE=2MB,
MAXSIZE=5MB,
FILEGROWTH=1MB
)
GO
```

（4）单击【执行】按钮执行输入的语句。如果执行成功，在查询窗口内的【查询】窗口中可以看到一条“命令已成功完成。”的消息。然后在【对象资源管理器】窗口中刷新，展开数据库节点就能看到刚创建的数据库了，如图 2-23 所示。

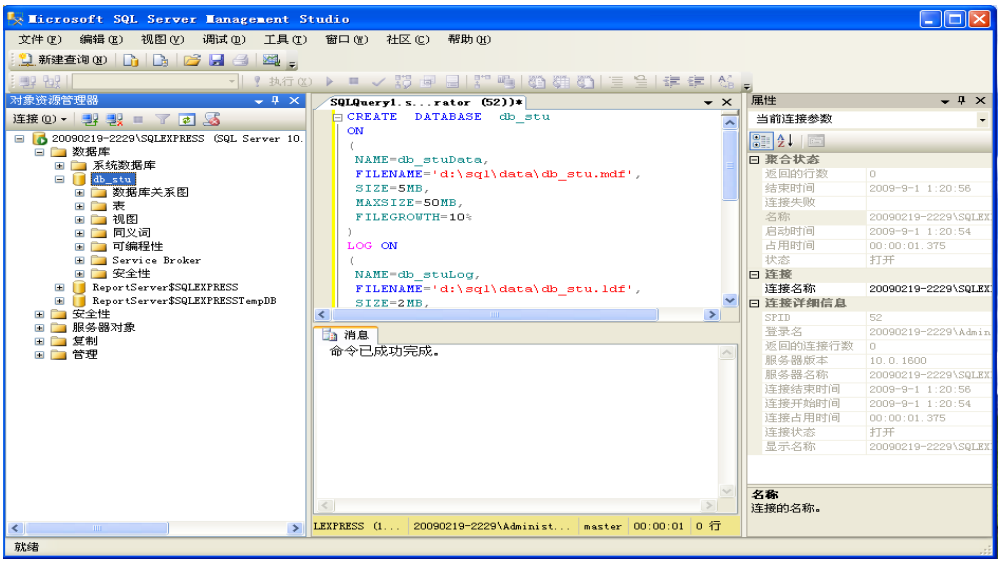


图 2-23　CREATE DATABSE 创建数据库

在上述的例子中，创建了【db_stu】数据库，其中 NAME 关键字指定了数据文件的逻辑名称是【db_stuData】，日志文件的逻辑名称是【db_stuLog】，物理名称是通过 FILENAME 关键字指定的。在【db_stu】数据库中，通过 SIZE 关键字把数据文件的大小设置为 5MB，最大值为 50MB，按 10%的比例增长；日志文件的大小设置为 2MB，最大值为 5MB，按 1MB 的方式增长。整个数据库的初始大小为数据文件大小（5MB）与日志文件大小（2MB）之和，即 7MB。

【例 2.2】 创建一个名为【TEST】的数据库，它有 3 个数据文件，其中主数据文件为 100MB，最大为 200MB，按 20MB 增长；两个辅助数据文件为 20M，并将这两个辅助数据文件存储在

名称为 group1 的文件组中，最大长度不限，按 10%增长；有两个日志文件，大小均为 50MB，最大均为 100MB，按 10MB 增长。

CREATE DATABASE 语句如下：

```
CREATE DATABASE TEST
ON    PRIMARY
(
NAME = 'TEST_data1',
FILENAME = 'e:\sql\data\test_data.ndf',
SIZE = 100MB,
MAXSIZE = 200MB,
FILEGROWTH = 20MB
),
FILEGROUP group1
(
NAME = 'TEST_data2',
FILENAME = 'e:\sql\data\test_data2.ndf',
SIZE = 20MB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 10%
),
(
NAME = 'TEST_data3',
FILENAME = 'e:\sql\data\test_data3.ndf',
SIZE = 20MB,
MAXSIZE = UNLIMITED,
FILEGROWTH = 10%
),
LOG ON
(
NAME = 'TEST_log1',
FILENAME = 'e:\sql\data\test_log1.ldf',
SIZE = 50MB,
MAXSIZE = 100MB,
FILEGROWTH = 10MB
),
(
NAME = 'TEST2_log',
FILENAME = 'e:\sql\data\test_log2.ldf',
SIZE = 50MB,
MAXSIZE = 100MB,
```

```
    FILEGROWTH = 10MB
    )
    GO
```

上述代码中，创建了 3 个数据文件和 2 个日志文件。如果数据库中的数据文件或日志文件多于 1 个，则文件之间使用逗号隔开。当数据文件有两个或两个以上时，需要指定哪一个数据文件是主数据文件。默认情况下，第一个文件就是主数据文件，也可以使用 PRIMARY 关键字来指定主数据文件。

2.3.2　使用 ALTER DATABASE 修改数据库

在 T-SQL 语句中，使用 ALTER DATABASE 语句修改数据库名称的语法形式如下：

```
    ALTER DATABASE old_databasename MODIFY NAME=new_databasename
```

【例 2.3】　将【db_stu】数据库名称修改为【学生信息管理系统】，语句如下：

```
    ALTER DATABASE db_stu MODIFY NAME=学生信息管理系统
```

执行完成后，即可在【对象资源管理器】窗口中右击数据库，选择【刷新】命令查看。

2.3.3　使用 DROP DATABASE 删除数据库

在 T-SQL 语句中，可以使用 DROP DATABASE 语句删除数据库，其语法如下：

```
    DROP DATABASE data_name [,……n];
```

其中，data_name 为要删除的数据库名，[,……n]表示可以有多于一个数据库名。

【例 2.4】　要删除数据库【db_stu】可使用如下的 DROP DATABASE 语句：

```
    DROP DATABASE db_stu
```

2.3.4　使用 CREATE TABLE 创建表

在 T-SQL 语句中，可以使用 CREATE TABLE 语句创建表，其语法格式如下：

```
    CREATE TABLE  [datebase_name.[owner].]table_name
    ({column_name data_type[NULL|NOT NULL]|[DEFAULT constant_expression]|[IDENTITY[(seed,increment)]]|[PRIMARY KEY|UQIQUE]}[,..n])[ON filegroup][TEXTIMAGE_ON filegroup]|[CONSTRAINT constraint_name]{CHECK[NOT FOR REPLICATION]}
```

其中，各参数的含义如下。

- [datebase_name.[owner].]table_name：定义表名称。
- column_name：定义列名。
- data_type：数据类型。
- NULL|NOT NULL：是否允许为空，默认为允许为空值。
- DEFAULT constant_expression：指定列的默认值。
- IDENTITY[(seed,increment)]：定义列为标志列。其中，seed 为标志种子，increment 为标志递增量，默认值为 1。

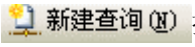
- PRIMARY KEY|UQIQUE：定义列为主关键字或 UNIQUE 约束。
- [,..n]：表示可以设计 n 个列的定义，每个列的定义用逗号隔开。
- ON filegroup：若用该项，表示将存储在 filegroup 制定的文件组中；若省略则表示存储在默认的文件组中。
- TEXTIMAGE_ON filegroup：用该项，表示表中若存在 TEXT 和 IMAGE 类型的列，则将这些类型的数据存储在 filegroup 指定的文件组中。
- CONSTRAINT constraint_name：定义约束名称，可以省略。
- CHECK：定义 check 约束。

使用 CREATE TABLE 语句创建表的具体操作步骤如下：

（1）在数据库【db_stu】中创建一个名字为【Course】的数据表。首先设计出表的结构如表 2-2 所示。

表 2-2　表【Course】的结构

字　段　名	数 据 类 型	长度（字节数）	完整性约束	中 文 描 述
Cno	char	5	NOT NULL	课程号
Cname	char	30	NOT NULL	课程名称
Ccredit	int	4	NULL	学分
Chours	int	4	NULL	学时数

（2）启动【Microsoft SOL Server Management Studio】，单击工具栏上的 

按钮，打开查询分析器，如图 2-24 所示。

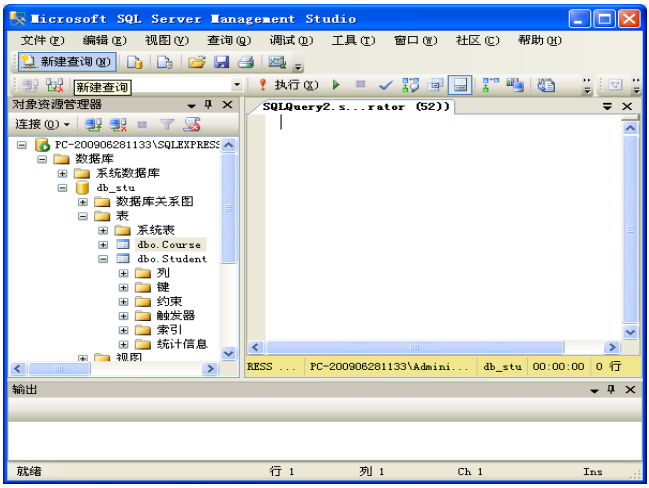


图 2-24　查询分析器界面

（3）在查询分析器中输入如下语句：

```
USE db_stu
CREATE TABLE Course
(Cno char(5)PRIMARY KEY NOT NULL,
Cname char(30)NOT NULL,
Ccredit int NULL,
```

```
    Chours int NULL
)
GO
```

（4）在工具栏中单击 执行(X) 按钮执行查询，完成数据表【Course】的创建，如图 2-25 所示。

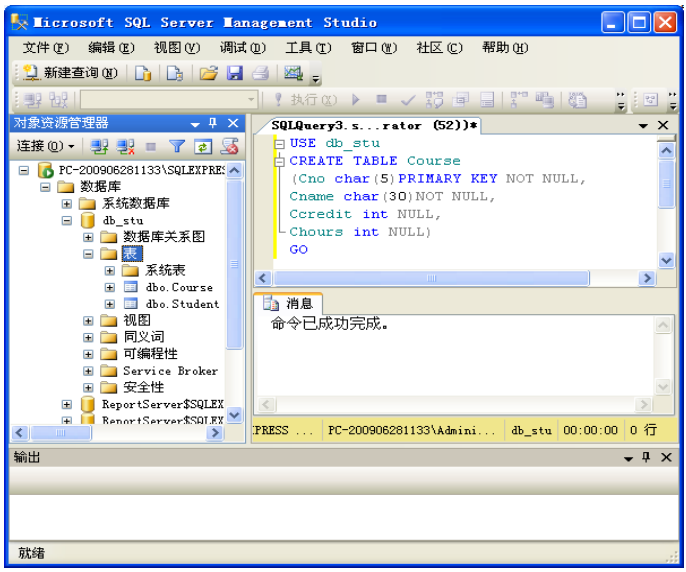


图 2-25　执行 T-SQL 语句结果

2.3.5　使用 ALTER TABLE 修改表

在 T-SQL 语句中，可以使用 ALTER TABLE 语句修改表，其语法格式如下：

```
ALTER TABLE table_name
{[ALTER COLUMN column_name{new_data_type[NULL|NOT NULL]}]|ADD column_name data_type[NULL|DEFAULT]|DROP COLUMN column_name[,...n]}
```

其中，各参数的含义如下。

- table_name：要修改的表的名称。
- ALTER COLUMN column_name：修改表中字段的属性，column_name 为字段名。
- ADD column_name data_type[NULL|DEFAULT]：添加字段，data_type 为该字段的数据类型，只允许增加可包含空值 NULL 或指定为 DEFAULT 的非空列。
- DROP COLUMN column_name[,...n]：表示要删除一个或多个字段。

【例 2.5】　用下面的语句修改表【Student】：

```
USE db_stu
ALTER TABLE Student
ADD Sdept char(10) NULL
```

单击 执行(X) 按钮执行 T-SQL 语句，为表【Student】添加【Sdept】字段。

2.3.6　使用 DROP TABLE 删除表

在 T-SQL 语句中可以使用 DROP TABLE 语句删除表，其语法格式如下：

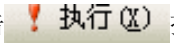
```
DROP TABLE table_name
```

其中，table_name 表示要删除的表的名字。

【例 2.6】　要删除数据库【db_stu】中的数据表【Student】，具体操作如下。

（1）在查询分析器中输入如下语句：

```
USE students
DROP TABLE Student
GO
```

（2）单击 

按钮执行 T-SQL 语句，删除表【Student】。

习　　题

1．表由_____和_____组成，字段对应的是_____，记录对应的是_____。

2．精确数字型数据类型包括_____、_____、_____和_____。

3．字符型数据是_____，包括_____、_____、_____和_____等。

4．在 T-SQL 语句中，可以使用_____语句创建表。

5．创建课程信息表【Course】，表结构如表 2-2 所示，将【课程号】定义为主关键字，并设置为表的标志列，标志初始值和增量均设置为 1。

6．用 T-SQL 语句创建一个数据库，在该数据库中创建一个表，输入数据，然后修改输入的数据，最后删除该数据表。

7．使用 SQL Server 管理控制台创建表【Student】，如表 2-3 所示。

表 2-3　表【Student】的结构

字　段　名	数 据 类 型	长度（字节数）	完整性约束	中 文 描 述
Sno	char	8	NOT NULL	学生学号
Sname	char	40	NOT NULL，UNIQUE	学生姓名
Ssex	char	1	NULL	学生性别
Sage	int	4	NULL	学生年龄
Sdept	char	10	NULL	学生系别

第 3 章　表数据操作

当数据库和表都创建完成以后，需要对表中的数据进行操作。对表中的数据操作包括插入、删除和修改数据，可以通过对象资源管理器窗口操作表中的数据，也可以通过 Transact-SQL 语句操作表中的数据。本章重点讨论操作表数据的两种方法。

3.1　界面操作表数据

以前面建立的数据库【db_stu】中的表【Student】为例，讲解如何在 SQL Serve 管理控制台中添加、修改和删除表【Student】中的记录。

3.1.1　插入记录

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【表】→【dbo.Student】，单击鼠标右键，在弹出的菜单中选择【编辑前 200 行】命令，在右边的窗口中会出现如图 3-1 所示的界面。

	Sno	Sname	Ssex	Sage	Sdept
*	NULL	NULL	NULL	NULL	NULL

图 3-1　添加记录前

（3）选中行内容都为 NULL 的记录，按照数据字典中所给的数据添加记录，添加完第一条记录以后，如图 3-2 所示。

	Sno	Sname	Ssex	Sage	Sdept
	50901001	WangYu ...	M	18	CS
▶*	NULL	NULL	NULL	NULL	NULL

图 3-2　添加第一条记录后

（4）按照第（3）步的做法，依次将表【Student】中的记录添加完成，然后关闭表【Student】，记录会自动保存。

（5）按照查看表中数据的步骤，查看表【Student】中的数据，如图 3-3 所示。

结果　消息

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901002	ZhangQiang	M	19	BA
3	50901003	HuangTing	F	18	FL
4	50901004	YeHuang	F	20	BA
5	50901005	XuYan	F	17	CS

图 3-3　所有记录添加完成

3.1.2　删除记录

如果想删除表中的记录时，可按以下步骤操作。

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【表】→【dbo.Student】，单击鼠标右键，在弹出的菜单中选择【编辑前 200 行】命令。

（3）在右边的窗口中用鼠标右键单击需要删除的记录，在弹出的菜单中选择【删除】命令，弹出提示对话框，如图 3-4 所示。

（4）单击【是】按钮，将记录永久删除。

图 3-4　删除数据时弹出的对话框

3.1.3　修改记录

当表【Student】中的数据添加完成以后，如果发现输入有错误可按以下步骤操作。

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【表】→【dbo.Student】，单击鼠标右键，在弹出的菜单中选择【编辑前 200 行】命令，在右边的窗口中找到要修改的学生记录，按照要求，将 19 改为 20。关闭表【Student】，修改后的记录会自动保存。

3.2　命令操作表数据

3.2.1　使用 INSERT 语句插入表数据

使用 INSERT 语句可以将一个或多个元组添加到表或视图中，语法格式如下：

```
INSERT
[INTO]
<table_name | view_name>
[(column_list)]
VALUES ({Default | Null | Expression }[,…n])
```

其中，参数说明如下。

- INTO：一个可选的关键字，可以将其用在 Insert 和目标表名或视图名之间。
- (column_list)：要在其中插入数据的一列或多列的列表，必须用括号将 column_list 括起来，并且用逗号表示分隔。
- VALUES：引入要插入的数据值的一个或多个列表。对于 column_list（如果已指定）或

表中的每个列，都必须有一个数据值。必须用圆括号将值列表括起来。

- Default：插入为列定义的默认值。如果某列并不存在默认值，并且该列允许 Null 值，则插入 Null。对于使用 timestamp 数据类型定义的列，插入下一个时间戳值。
- Expression：一个常量、变量或表达式，其值的数据类型要与列的数据类型一致。注意表达式不能包含 EXECUTE 语句。

用户往一张表中插入数据时，由于某些列的值不能确定，只能先插入其他列的值，这种情况下，column_list 属性列名表的参数一定不能省略，要指出为哪些列插入值，并且 VALUES 后面引导的值的顺序必须与 column_list 的属性列名表的顺序一致。

例如，确定开某一门课程时，它的课程号及课程的名称我们可以确定，但是对于具体的学分及学时数可能还需要通过教研组再商讨，所以只能插入 Null 值。

【例 3.1】　在课程表中，插入一条记录，但是学分和学时未知。

```
INSERT　INTO　Course( Cno , Cname )
VALUES ( '5' , 'J2SE' )
```

运行结果如图 3-5 所示。

然后查看表【Course】，插入的数值如图 3-6 所示，可以看出，【Ccredit】列和【Chours】列的值为 Null。

图 3-5　插入一行数据后的消息提示

	Cno	Cname	Ccredit	Chours
1	1	DB_Design	2	36
2	2	VB_Programming	2	36
3	3	MIS	4	72
4	4	BIT	3	54
5	5	J2SE	NULL	NULL

图 3-6　插入数据后的表【Course】

设计表时，如果一张表的某个列不允许为空，在插入数据时，必须为这些列赋值，否则会导致错误。我们来看下面的例子。

```
INSERT　INTO　Student( Ssex , Sage )
VALUES ( 'F' , 21 )
```

运行结果如图 3-7 所示。

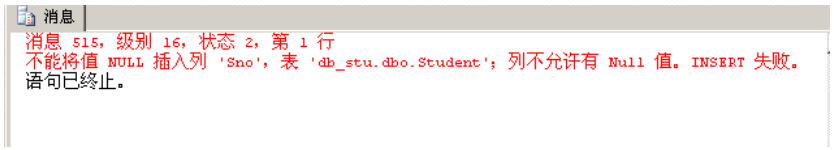


图 3-7　消息 515

从图 3-7 中可以看出，插入数据失败，原因就是没有给【Sno】列插入数据，而在设计表【Student】时，【Sno】列不能为空。

有些时候，用户也会往一张表中插入一行完整的数据，即为每一个属性列都赋值。这时 column_list 属性列名表的参数有两种使用情况：一种是列出所有的属性列的名称，另一种是省略属性列名表。

【例 3.2】　为学生表插入两条完整的记录后并查看。

```
INSERT　INTO　Student
VALUES ( '50901006' , 'TangXin' ,　'M' , 19 ,　'FL' ),
　　( '50901007' , 'ChenLi' , 'F' , 20 , 'FL')
```

运行结果如图 3-8 所示。

然后查看表【Student】，插入的数值如图 3-9 所示。

图 3-8　插入两行数据后的消息提示

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901002	ZhangQiang	M	19	BA
3	50901003	HuangTing	F	18	FL
4	50901004	YeHuang	F	20	BA
5	50901005	XuYan	F	17	CS
6	50901006	TangXin	M	19	FL
7	50901007	ChenLi	F	20	FL

图 3-9　插入数据后的表【Student】

3.2.2　使用 DELETE 或 TRANCATE 语句删除数据

1．DELETE 语句

数据表使用一段时间以后，会产生大量的冗余数据，这不但增加了硬件开销，也给数据的维护带来了很多麻烦。比如，学生毕业以后，他所使用的校园一卡通就不能够再使用了，这时就需要对他的一卡通号码予以注销。

DELETE 语句用于从表或视图中删除行，可以根据 WHERE 后面条件的设置，每次从表或视图中删除一行或多行。语法格式如下：

```
DELETE　[FROM]　< table_name | view_name >
[WHERE< search_condition >]
```

【例 3.3】　删除学生 TangXin 的信息。

```
DELETE　FROM　Student
WHERE　Sname =　' TangXin '
```

【例 3.4】　删除【SC】表的所有记录。

```
DELETE　FROM　SC
```

2．TRUNCATE 语句

TRUNCATE 语句用于删除表中的所有行。TRUNCATE　TABLE 与没有 WHERE 子句的 DELETE 语句类似。语法格式如下：

```
TRUNCATE　　TABLE　<table_name>
```

例 3.4 也可以用下面的 SQL 语句完成，执行结果相同：

```
TRUNCATE　TABLE　　SC
```

3.2.3　使用 UPDATE 语句修改数据

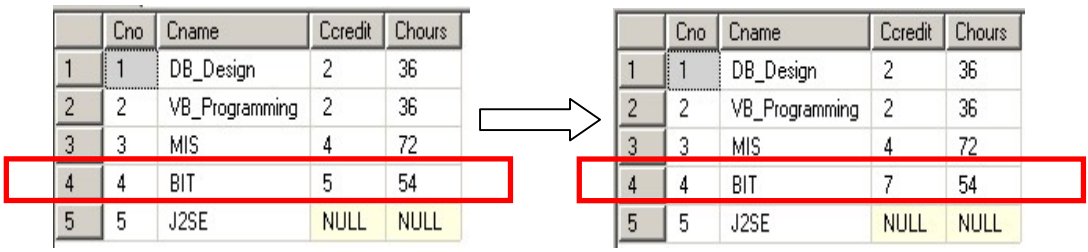
UPDATE 语句用于更改表或视图中的现有数据，语法格式如下：

```
UPDATE    < table_name | view_name >
SET      <column_name1> = <expression1>
        [<column_name2> = <expression2>]
……
[WHERE < search_condition >]
```

【例 3.5】　将 BIT 课程的学分增加 2。

```
UPDATE    Course
SET    Ccredit = Ccredit +2
WHERE    Cname = 'BIT'
```

运行的结果如图 3-10 所示。



	Cno	Cname	Ccredit	Chours
1	1	DB_Design	2	36
2	2	VB_Programming	2	36
3	3	MIS	4	72
4	4	BIT	5	54
5	5	J2SE	NULL	NULL

	Cno	Cname	Ccredit	Chours
1	1	DB_Design	2	36
2	2	VB_Programming	2	36
3	3	MIS	4	72
4	4	BIT	7	54
5	5	J2SE	NULL	NULL

图3-10　修改前的结果和修改后的结果对比

习　　题

1. 在 SQL Server 2008 的对象资源管理器窗口中对数据进行修改，与使用 Transact-SQL 语句修改数据，两种方法相比较，哪一种功能更强大、更灵活？试举例说明。

2. 写出 Transact-SQL 语句，对表【SC】进行如下操作。

（1）插入数据，将 ZhangSan 的 VB_Programming 成绩 80 分插入到【SC】表中。

（2）修改数据，将 ZhangSan 的 VB_Programming 成绩更正为 85 分。

（3）删除数据，删除 ZhangSan 的 VB_Programming 成绩。

第 4 章　数据库的查询和视图

本章将详细讲解 SQL Server 2008 中的 SQL 查询技术。通过对本章的学习，可掌握多种查询方法，包括单表、多表查询，多条件查询，嵌套查询，并能对查询结果进行分组、排序等操作。

视图对于数据库的用户来说非常重要，它是由一个或多个基本表导出的数据信息，可以根据用户的需要创建视图。本章将介绍视图的概念，以及视图的创建与使用方法。

游标在数据库管理系统（DBMS）与应用程序之间提供了数据处理单位的变换机制，本章还将讨论游标的概念及使用方法。

4.1　连接、选择和投影

SQL Server 2008 是一种关系数据库管理系统，在关系数据库中，必须提供一种对二维表进行运算的机制。这种机制除包括传统的集合运算中的并、交、差、广义笛卡儿积以外，还包括专门的关系运算中的选择、投影和连接。

4.1.1　选择（Selection）

选择是单目运算，它是按照一定的条件，从关系 R 中选择出满足条件的行作为结果返回。选择运算的操作对象是一张二维表，其运算结果也是一张二维表。

选择运算可以记做 $\sigma_F(R)$，其中σ是选择运算符，F 表示选择的条件，R 表示被操作的二维表。

【例 4.1】　对年龄为 18 岁的学生信息进行查询，可以表示为：

$$\sigma_{\text{Sage=18}}(\text{Student}) \text{ 或 } \sigma_{4=18}(\text{Student})$$

结果如表 4-1 所示。

表 4-1　查询结果一

Sno	Sname	Ssex	Sage	Sdept
50901001	WangYu	M	18	CS
50901003	HuangTing	F	18	FL

【例 4.2】　对年龄为 18 岁的女生信息进行查询，可以表示为：

$$\sigma_{\text{Sage=18}\wedge\text{Ssex='F'}}(\text{Student}) \text{ 或 } \sigma_{4=18\wedge 3='F'}(\text{Student})$$

结果如表 4-2 所示。

表 4-2　查询结果二

Sno	Sname	Ssex	Sage	Sdept
50901003	HuangTing	F	18	FL

4.1.2　投影（Projection）

投影是单目运算，它是将二维表 *R* 的所有元组去掉某些分量后，形成的一张新二维表。

投影运算可以记做 $\Pi_A(R)$，其中 Π 是投影运算符，*A* 表示结果表中包含的属性列的名称，*R* 表示被操作的二维表。

【例 4.3】　对学生表的学号和姓名进行投影，可以表示为：

$$\Pi_{\text{Sno,Sname}}(\text{Student})$$

结果如表 4-3 所示。

表 4-3　投影结果一

Sno	Sname
50901001	WangYu
50901002	ZhangQiang
50901003	HuangTing
50901004	YeHuang
50901005	XuYan

在例 4-3 中，做完投影运算以后，对于学生表【Student】而言，由原来的 5 列变成了 2 列，相当于取消了原表中的 3 个属性列。有些情况下，还可能会取消原关系中的某些元组。

【例 4.4】　对学生表的系别进行投影，可以表示为：

$$\Pi_{\text{Sdept}}(\text{Student})$$

结果如表 4-4 所示。

表 4-4　投影结果二

Sdept
CS
BA
FL

从表 4-4 中可以看出，学生表【Student】中原来有 5 个元组，做完投影运算以后，结果表中只有 3 个元组。这是因为，去掉了学生表【Student】中的 Sno、Sname、Ssex、Sage4 个属性列以后，出现了重复行。我们知道，在关系表中要求任意两个元组不能完全相同，所以应该去掉这些重复行。

4.1.3　连接（Join）

连接运算就是将两张二维表按照给定的条件进行拼接后形成一张新的二维表。

连接运算可以记做 $R \underset{F}{\infty} S$，其中 *R* 和 *S* 表示被操作的两张二维表，*F* 表示给定的条件。

【例 4.5】　如果表【R】和表【S】分别如表 4-5 和表 4-6 所示，则 $\mathrm{R} \underset{\mathrm{C<E}}{\infty} \mathrm{S}$ 的结果如表 4-7 所示。

表 4-5　【R】表

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	11

表 4-6　【S】表

B	E
b1	4
b2	7
b3	10
b3	2
b5	2

表 4-7　连接结果一

A	R.B	C	S.B	E
a1	b1	5	b2	7
a1	b1	5	b3	10
a1	b2	6	b2	7
a1	b2	6	b3	10
a2	b3	8	b3	10

当连接的条件是两张表的某些属性列的值相等时，我们称之为等值连接，等值连接也是连接运算中较为常用的一种连接方式。

【例 4.6】　如果表【R】和表【S】分别如表 4-5 和表 4-6 所示，则 $R \underset{r.b=s.b}{\infty} S$ 的结果如表 4-8 所示。

表 4-8　连接结果二

A	R.B	C	S.B	E
a1	b1	5	b1	4
a1	b2	6	b2	7
a2	b3	8	b3	10
a2	b3	8	b3	2

等值连接中还有一种特殊的连接方式称为自然连接。自然连接要求两张二维表中相等的分量必须是相同的属性组，并且在结果表中把相同的属性列去掉。自然连接运算可以记做 $R \infty S$ 。

【例 4.7】　如果表【R】和表【S】分别如表 4-5 和表 4-6 所示，则 $R \infty S$ 的结果如表 4-9 所示。

从表 4-8 和表 4-9 可以看出，自然连接实际上可以看做是做完等值连接运算以后，再将相同的属性列合并的操作。

表 4-9　自然连接结果

A	B	C	E
a1	b1	5	4
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2

4.2　数据库的查询

数据库的使用为我们带来了极大的方便，它将数据进行存储，并且在用户需要时随时提供相关的数据信息。因此可以看出，数据库的核心操作实际上就是对数据进行查询。

SELECT 子句是数据查询的核心内容。它不仅可以以不同的方式查询数据库中的数据，而且可以查询出经过计算的数据库中的数据，如学生某门课程考试成绩的最高分、最低分、平均分等，其功能非常强大。

虽然查询语句的完整语法比较复杂，但是我们可以将其中的主要子句的语法格式归纳如下：

```
SELECT  select_list  [INTO  new_table]
[GROUP  BY  group_by_expression]
[HAVING  search_condition]
[ORDER  BY  order_expression [ASC | DESC]]
```

4.2.1　选择列

1．查询部分字段

用 SELECT 子句选择表中的字段时，只需将希望结果表中显示的字段名称置于 SELECT 子句后即可，多个字段名称之间用逗号分隔。

【例 4.8】　查询课程的课程名称、学分和学时。

```
SELECT  Cname,Ccredit,Chours
FROM  Course
```

单击 新建查询(N) 按钮，先将数据库名称改为 db_stu，如图 4-1 所示。然后在右边的空白处输入例 4.8 中的 SQL 语句，运行结果会在下方显示，如图 4-2 所示。

2．查询全部字段

如果要显示某张表中所有字段的值，可以将所有字段名称列于 SELECT 子句后面，也可以用*代替所有的字段名称。

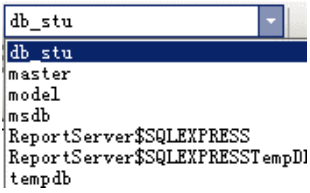


图 4-1　修改数据库名称为【db_stu】

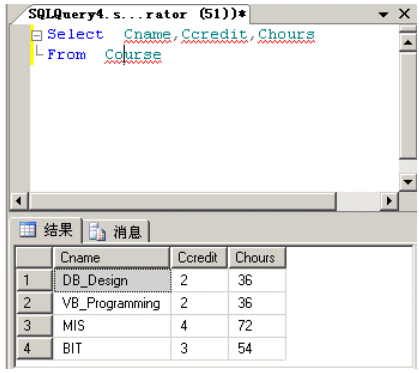


图 4-2　SELECT 子句查询结果一

【例 4.9】　查询课程的详细信息。

```
SELECT　*
FROM　Course
```

运行结果如图 4-3 所示。

3．查询经过计算的值

SELECT 子句后的 select_list 不仅可以是字段名称，还可以是一个算术表达式，或者是一个函数。

【例 4.10】　查询学生的姓名和出生年份。

```
SELECT　Sname,2010 – Sage
FROM　Student
```

运行结果如图 4-4 所示。

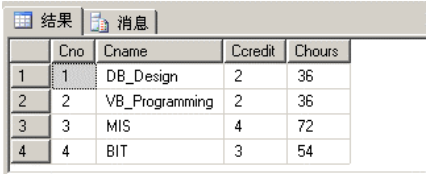


图 4-3　SELECT 子句查询结果二

图 4-4　SELECT 子句查询结果三

从图 4-4 中可以看出，经过计算的列是没有列名的，如果要加上列名出生日期，可以修改例 4-10 中的 SQL 语句如下：

```
SELECT　Sname,2010 – Sage　出生日期
FROM　Student
```

结果如图 4-5 所示。

【例 4.11】　查询学生课程号为 4 的课程的考试成绩的平均分。

```
SELECT　AVG(Grade)
FROM　SC
WHERE　Cno =　'4'
```

运行结果如图 4-6 所示。

	Sname	出生日期
1	WangYu	1992
2	ZhangQiang	1991
3	HuangTing	1992
4	YeHuang	1990
5	XuYan	1993

图 4-5　SELECT 子句查询结果三加上列名后的查询结果

	(无列名)
1	84

图 4-6　SELECT 子句查询结果四

【例 4.12】　查询学生的总人数。

```
SELECT　COUNT(*)
FROM　Student
```

运行结果如图 4-7 所示。

4．查询空值

实际应用中，空值表示还没有向数据库中添加数据，它往往有两种含义：第一种是其值未知，如学生选完课后，但还未进行期末考试前，该门课程的成绩是未知的；第二种是其值不存在，如学生某门课程旷考，那么该门课程的成绩就不存在，也可以用空值表示。

【例 4.13】　查询【SC】表没有成绩的学生的学号和课程号。

```
SELECT　Sno,Cno
FROM　SC
WHERE　Grade　IS　NULL
```

运行结果如图 4-8 所示。

	(无列名)
1	5

图 4-7　SELECT 子句查询结果五

	Sno	Cno
1	50901001	1

图 4-8　SELECT 子句查询结果六

5．DISTINCT 关键字

DISTINCT 关键字用于去除查询结果中的重复元组。例如，查询课程的学分值有哪些，可以使用如下的 SELECT 子句。

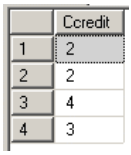
```
SELECT　Ccredit
FROM　Course
```

结果如图 4-9 所示。

由图 4-9 可以看出，因为 DB_Design 课程和 VB_Programming 课程的学分都为 2，所以图中第 1 个元组和第 2 个元组的值都为 2，如果要在课程学分值的列表中去掉这个重复的值，可以使用下面的 SELECT 子句：

```
SELECT　DISTINCT　Ccredit
FROM　Course
```

运行结果如图 4-10 所示。



	Ccredit
1	2
2	2
3	4
4	3

图 4-9　SELECT 子句查询结果七

	Ccredit
1	2
2	3
3	4

图 4-10　SELECT 子句查询结果八

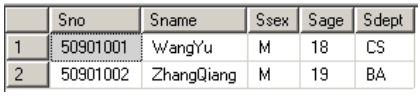
6．TOP 关键字

TOP 关键字可以限制查询结果显示的元组的个数。

【例 4.14】　查询学生表中的前 2 条记录。

```
SELECT　TOP　2　*
FROM　Student
```

运行结果如图 4-11 所示。



	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901002	ZhangQiang	M	19	BA

图 4-11　SELECT 子句查询结果九

【例 4.15】　查询学生表中的前 20%的记录。

```
SELECT　TOP 20 PERCENT *
FROM　Student
```

运行结果如图 4-12 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS

图 4-12　SELECT 子句查询结果十

7．INTO 子句

INTO 子句可以在默认的字段组中创建一个新表，并将来自查询的结果表的元组插入新表中，常用来给数据表建立备份。

【例 4.16】　为学生表建立一个备份。

```
SELECT　*　INTO　Student_backup
FROM　Student
```

4.2.2　选择行

一张数据表中存储的数据通常有几万条甚至更多，而用户在查询时往往只关心其中的一部分记录，这时就需要使用 WHERE 子句指定一系列的查询条件。

1．使用比较运算符的查询

常用的比较运算符主要包括>（大于）、>=（大于等于）、<（小于）、<=（小于等于）、=（等于），以及!=和<>（不等于）等。

【例 4.17】　查询学号为 50901004 的学生的详细记录。

```
SELECT　*
FROM　Student
WHERE　Sno =　'50901004 '
```

运行结果如图 4-13 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901004	YeHuang	F	20	BA

图 4-13　WHERE 子句查询结果一

【例 4.18】　查询 3 个学分以上的课程的名称。

```
SELECT　Cname
FROM　Course
WHERE　Ccredit >= 3
```

运行结果如图 4-14 所示。

2．使用 LIKE 关键字的查询

用户有时对要查询的数据表中的数据了解不是很全面，如要查找某个学生的信息，但是不知道该学生的姓氏，只知道学生名字叫 Yan，这时就需要使用 LIKE 关键字进行模糊查询。LIKE 关键字需要使用通配符在字符串内查找指定的模式，所以我们先来了解通配符的含义。常用的通配符如表 4-10 所示。

	Cname
1	MIS
2	BIT

图 4-14　WHERE 子句查询结果二

表 4-10　通配符列表

通　配　符	含　　义
_	匹配任意的单个字符
%	匹配任意的长度大于等于零的字符串
[ ]	匹配特定范围的任意单个字符
[^]	匹配特定范围之外的任意单个字符

【例 4.19】　查询名字叫 Yan 的学生的学号和姓名。

```
SELECT　Sno,Sname
FROM　Student
WHERE　Sname　LIKE　'%Yan '
```

运行结果如图 4-15 所示。

	Sno	Sname
1	50901005	XuYan

图 4-15　WHERE 子句查询结果三

【例 4.20】　查询课程名称中的第二个字为 I 且长度为 3 的课程名称。

```
SELECT　Cname
FROM　Course
WHERE　Cname　LIKE　'_I_ '
```

运行结果如图 4-16 所示。

对于本书中表【Course】中的第一条记录，使用模糊查询时，查询课程名为 DB_Design 的课程信息，如果直接写成 LIKE 'XX_XXXXXX'，则会把“_”当成是 LIKE 的通配符去处理。SQL 里提供了 ESCAPE 子句来处理这种情况，我们可以用 ESCAPE 指定 LIKE 中使用的转义符是什么。这样，跟在转义符后面的字符就会被当成原始字符去处理。

【例 4.21】　查询课程名称中含有“DB_”的课程详细信息。

```
SELECT　*
FROM　Course
WHERE　Cname　LIKE　'DB!_%'
ESCAPE　'!'
```

运行结果如图 4-17 所示。

	Cname
1	MIS
2	BIT

图 4-16　WHERE 子句查询结果四

	Cno	Cname	Ccredit	Chours
1	1	DB_Design	2	36

图 4-17　WHERE 子句查询结果五

3．查询同时满足多个条件的元组

逻辑运算符 AND 可返回满足所有条件的元组。

【例 4.22】　查询系别为 CS 的男生的姓名。

```
SELECT　Sname
FROM　Student
WHERE　Sdept =　'CS '　AND　Ssex =　'M '
```

运行结果如图 4-18 所示。

4．查询满足若干个条件中任意一个条件的元组

逻辑运算符 OR 可返回满足任意一个条件的元组。

【例 4.23】　查询系别为 CS 或者年龄等于 18 岁的学生的详细信息。

```
SELECT　*
FROM　Student
WHERE　Sdept =　'CS '　OR　Sage = 18
```

运行结果如图 4-19 所示。

	Sname
1	WangYu

图 4-18　WHERE 子句查询结果六

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901003	HuangTing	F	18	FL
3	50901005	XuYan	F	17	CS

图 4-19　WHERE 子句查询结果七

5．查询含有（或不含有）两个值之间的任意一个值的元组

BETWEEN X AND Y 和 NOT BETWEEN X AND Y 可返回属性值在或不在值 X 和值 Y 之间的元组，使用时需要注意的是，值 Y 应该大于值 X。

【例 4.24】　查询学时数在 36 到 54 之间的课程的名称及学时数。

```
SELECT　Cname,Chours
FROM　Course
WHERE　Chours　BETWEEN　36 and　54
```

运行结果如图 4-20 所示。

上面的程序等价于：

```
SELECT　Cname,Chours
FROM　Course
WHERE　Chours >= 36　AND　Chours <= 54
```

注意：返回的结果中包括值 36，也包括值 54。

【例 4.25】　查询年龄不在 19 到 20 岁之间的学生的信息。

```
SELECT　*
FROM　Student
WHERE　Sage　NOT　BETWEEN　19　and　20
```

运行结果如图 4-21 所示。

	Cname	Chours
1	DB_Design	36
2	VB_Programming	36
3	BIT	54

图 4-20　WHERE 子句查询结果八

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901003	HuangTing	F	18	FL
3	50901005	XuYan	F	17	CS

图 4-21　WHERE 子句查询结果九

注意：返回的结果中不包括值 19，也不包括值 20。

6．查询列与值列表中的元组

通常使用关键字 IN 来指定列表搜索的条件，查询属性值属于指定集合的元组。IN 关键字的使用格式是：IN（expression 1,expression 2, expression 3,……）。

【例 4.26】　查询学号为 50901002、50901003 和 50901004 的学生的学号和姓名。

```
SELECT　Sno,Sname
FROM　Student
Sno　　IN ('50901002', '50901003' , '50901004' )
```

运行结果如图 4-22 所示。

IN 关键字也可以和 NOT 配合起来使用，用于查询属性值不属于指定集合的元组。

【例 4.27】　查询学号不为 50901002、50901003 和 50901004 的学生的学号和姓名。

```
SELECT　Sno,Sname
FROM　Student
```

```
WHERE　Sno　NOT　IN ('50901002' , '50901003' , '50901004' )
```

运行结果如图 4-23 所示。

	Sno	Sname
1	50901002	ZhangQiang
2	50901003	HuangTing
3	50901004	YeHuang

图 4-22　WHERE 子句查询结果十

	Sno	Sname
1	50901001	WangYu
2	50901005	XuYan

图 4-23　WHERE 子句查询结果十一

7．嵌套查询

一个外层查询中包含有另一个内层查询称为嵌套查询，即一个 SELECT...FROM...WHERE 查询语句块可以嵌套在另一个 SELECT...FROM...WHERE 查询语句块的 WHERE 子句中。其中，外层查询称为主查询，内层查询称为子查询，子查询的结果作为主查询的条件。

1）含有 IN 的嵌套查询

【例 4.28】　查询与 WangYu 同龄的学生的姓名。

```
SELECT　Sname
FROM　Student
WHERE　Sage　IN
( SELECT　Sage
FROM　Student
WHERE　Sname =　'WangYu '
)
```

	Sname
1	WangYu
2	HuangTing

图 4-24　嵌套查询结果一

运行结果如图 4-24 所示。

说明：当子查询的返回值为 1 时，in 可以换成“＝”。

【例 4.29】　查询选修了 3 号课程的学生的姓名。

```
SELECT　Sname
FROM　Student
WHERE　Sno　IN
( SELECT　Sno
FROM　SC
WHERE　Cno =　'3'
)
```

运行结果如图 4-25 所示。

【例 4.30】　查询选修了 MIS 课程的学生的姓名。

```
SELECT　Sname
FROM　　Student
WHERE　Sno　IN
( SELECT　Sno
FROM　SC
WHERE　Cno　IN
( SELECT　Cno
```

```
FROM    Course
WHERE    Cname =    'MIS '
)
)
```

运行结果如图 4-26 所示。

	Sname
1	WangYu
2	ZhangQiang

图 4-25　嵌套查询结果二

	Sname
1	WangYu
2	ZhangQiang

图 4-26　嵌套查询结果三

2）含有 ANY 和 ALL 的嵌套查询

如表 4-11 所示为含有 ANY 和 ALL 的查询的含义。

表 4.11　含有 ANY 和 ALL 的查询的含义

查　　询	含　　义
>ALL	大于子查询结果中的所有值
<ALL	小于子查询结果中的所有值
>ANY	大于子查询结果中的某个值
<ANY	小于子查询结果中的某个值

【例 4.31】　查询其他系中比 BA 系所有学生年龄都小的学生的详细信息。

```
SELECT    *
FROM    Student
WHERE    Sage < ALL
  ( SELECT     Sage
FROM    Student
WHERE    Sdept =    'BA '
)
AND    Sdept != 'BA'
```

运行结果如图 4-27 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901003	HuangTing	F	18	FL
3	50901005	XuYan	F	17	CS

图 4-27　嵌套查询结果四

【例 4.32】　查询其他系中比 CS 系某一学生年龄大的学生的详细信息。

```
SELECT    *
FROM    Student
WHERE    Sage > ANY
```

```
( SELECT    Sage
FROM    Student
WHERE    Sdept =    'CS '
)
AND    Sdept != 'CS '
```

运行结果如图 4-28 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901002	ZhangQiang	M	19	BA
2	50901003	HuangTing	F	18	FL
3	50901004	YeHuang	F	20	BA

图 4-28　嵌套查询结果五

4.2.3　FROM 子句

FROM 子句指定了在 SELECT、UPDATE、DELETE 子句中使用的数据源的构成形式，后面可以跟要查询的表或者视图等的名称。

对于 FROM 子句后跟查询表的形式，从前面的例子中，读者应该已经了解了它的作用。我们也可以使用 AS 关键字给所要查询的表指定别名。AS 关键字也可以省略，直接给出别名。别名形式的使用在表的自身连接查询中应用相对多一些。

FROM 子句后还可以跟视图名，也可以使用 AS 关键字或省略的形式为其指定别名。有关视图的介绍请参阅 4.3 节。

4.2.4　连接

连接查询就是对两张或者两张以上的表进行查询。使用时，将涉及的表的名称写在 FROM 子句的后面，表名与表名之间用逗号进行分隔，任意两张表的连接条件分别写在 WHERE 子句的后面。

【例 4.33】　查询 ZhangQiang 同学 BIT 课程的考试成绩。

```
SELECT    Grade
FROM    Student , Course , SC
WHERE    Student.Sno = SC.Sno    AND    Course.Cno = SC.Cno    AND
Sname = 'ZhangQiang '    AND    Cname =    'BIT '
```

运行结果如图 4-29 所示。

	Grade
1	90

图 4-29　连接查询结果

4.2.5　数据汇总

1．GROUP BY 子句

GROUP BY 子句可以将数据表的元组按照一个或多个字段划分为不同的组。针对每一组

返回一行。

【例 4.34】　对于学生表，按照性别进行分组。

```
SELECT　Ssex
FROM　Student
GROUP　BY　Ssex
```

运行结果如图 4-30 所示。

	Ssex
1	F
2	M

图 4-30　GROUP BY 子句查询结果一

需要注意的是，在 SELECT 子句的字段列表中，除了聚合函数之外，其他出现的字段一定要在 GROUP BY 子句中出现才可以。读者可以运行下面的 SQL 语句，运行结果如图 4-31 所示。

```
SELECT　Sname,Ssex
FROM　Student
GROUP　BY　Ssex
```

消息

消息 8120，级别 16，状态 1，第 1 行
选择列表中的列 'Student.Sname' 无效，因为该列没有包含在聚合函数或 GROUP BY 子句中。

图 4-31　消息 8120

【例 4.35】　对于学生表【Student】，请按照年龄和性别进行分组。

```
SELECT　Sage,Ssex
FROM　Student
GROUP　BY　Sage,Ssex
```

运行结果如图 4-32 所示。

2．HAVING 子句

前面学习过 WHERE 子句，它主要和关键字 SELECT 配合使用，用于分组之间的条件，而分组之后的条件就要用到 HAVING 子句了，它主要和 GROUP BY 配合使用。

【例 4.36】　查询选修 3 门以上课程的学生的学号。

```
SELECT　Sno
FROM　SC
GROUP　BY　Sno
HAVING　COUNT ( * ) >= 3
```

运行结果如图 4-33 所示。

	Sage	Ssex
1	17	F
2	18	F
3	18	M
4	19	M
5	20	F

图 4-32　GROUP BY 子句查询结果二

	Sno
1	50901001

图 4-33　HAVING 子句查询结果一

【例 4.37】　查询至少有两个学生选修的课程号。

	Cno
1	3
2	4

图 4-34　HAVING 子句查询结果二

```
SELECT　Cno
FROM　SC
GROUP　BY　Cno
HAVING　COUNT ( * ) >= 2
```

运行结果如图 4-34 所示。

4.2.6　排序

ORDER BY 子句可以对查询结果按照一个或多个属性列的升序（ASC）或降序（DESC）排列，默认值为升序排列。

【例 4.38】　查询学生的详细信息，查询结果按照学生姓名进行降序排列。

```
SELECT　*
FROM　Student
ORDER　BY　Sname　DESC
```

运行结果如图 4-35 所示。

【例 4.39】　查询选课表的详细信息，查询结果按照成绩进行升序排列。

```
SELECT　*
FROM　SC
ORDER　BY　Grade　ASC
```

运行结果如图 4-36 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901002	ZhangQiang	M	19	BA
2	50901004	YeHuang	F	20	BA
3	50901005	XuYan	F	17	CS
4	50901001	WangYu	M	18	CS
5	50901003	HuangTing	F	18	FL

图 4-35　ORDER BY 子句查询结果一

	Sno	Cno	Grade
1	50901001	1	NULL
2	50901001	4	78
3	50901002	3	80
4	50901001	3	80
5	50901002	4	90

图 4-36　ORDER BY 子句查询结果二

说明：从图 4.36 的排序结果中可以看出，在 SQL Server 中，空值被当做最小值处理。

【例 4.40】　查询学生的详细信息，查询结果按照学生系别进行升序排列，同一系中的学生按照学生的姓名进行降序排列。

```
SELECT　*
FROM　Student
ORDER　BY　Sdept, Sname　DESC
```

运行结果如图 4-37 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901002	ZhangQiang	M	19	BA
2	50901004	YeHuang	F	20	BA
3	50901005	XuYan	F	17	CS
4	50901001	WangYu	M	18	CS
5	50901003	HuangTing	F	18	FL

图 4-37　ORDER BY 子句查询结果三

4.3　视　　图

在关系数据库中，为用户提供多种角度观察数据的重要机制就是视图。对于用户来说，视图就像一个窗口，不同的用户从不同的窗口去看数据库，看到的数据是不一样的。这就犹如画图中提到的视图的概念，对于同样的物体，从不同的角度去看，画出的图自然不同，但画的却都是同一种物体。

4.3.1　视图概念

视图是一张虚拟表，是从一张或多张表（或视图）中导出的表，实际上是一个查询结果。它并不表示任何的物理数据，也就是说，视图所对应的数据不会进行实际存储，数据库中只存储了视图的定义，这些数据仍然存储在导出视图的基本表中。当基本表中的数据发生改变时，通过视图查询出来的数据也会随之改变。同样，如果对通过视图看到的数据进行修改时，相应的基本表中的数据也会随之改变。

视图定义完成以后，就可以像基本表一样被查询和更新了。视图的作用可以归纳为以下几点。

（1）简化了用户操作数据的方式。可以为复杂的查询建立一个视图，这样，用户每次查询时，就不用输入复杂的查询语句，只需对建立的视图做简单的查询即可。

（2）对不同的用户建立不同的视图，使用户只能看到与自己相关的数据。这不仅简化了用户权限的管理，而且增加了安全性。

（3）对不同的用户来说，自己所需要的数据不需要都定义和存储，可以共享数据库中的数据。

4.3.2　创建视图

创建视图有两种方法：一种是通过 SQL Server Management Studio 的对象资源管理器创建视图，另一种是使用 Transact-SQL 的 CREATE VIEW 语句创建视图。

1．在 SQL Server Management Studio 的对象资源管理器中创建视图

以在【db_stu】数据库中创建【CS_STU】（描述计算机专业的学生情况）视图为例说明在对象资源管理器窗口中创建视图的过程。

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【视图】，单击鼠标右键，出现如图 4-38 所示的界面，选择【新建视图】命令。

图 4-38　【新建视图】命令

（3）在弹出的如图 4-39 所示的【添加表】对话框中选择与创建视图相关联的表、视图、函数或者同义词。可以使用【Ctrl】键或者【Shift】键进行多选。选择完成后，单击【添加】按钮，如图 4-40 所示。

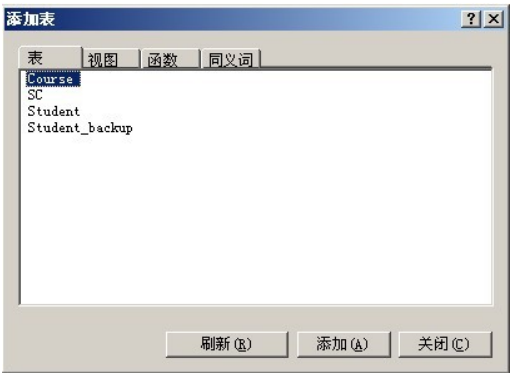


图 4-39　【添加表】对话框

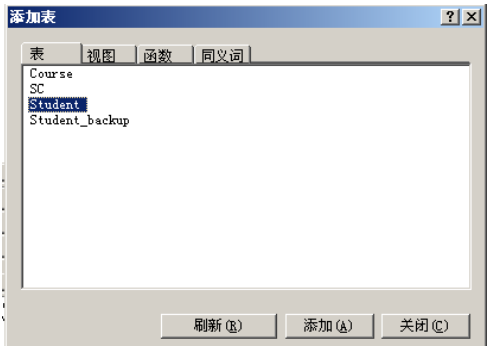


图 4-40　选择基本表

在如图 4-41 所示的窗口中选择创建视图所需的字段，可以指定列的别名、排序类型、排序顺序和筛选条件等。本例中指定 Sdept 字段的筛选条件为 CS。

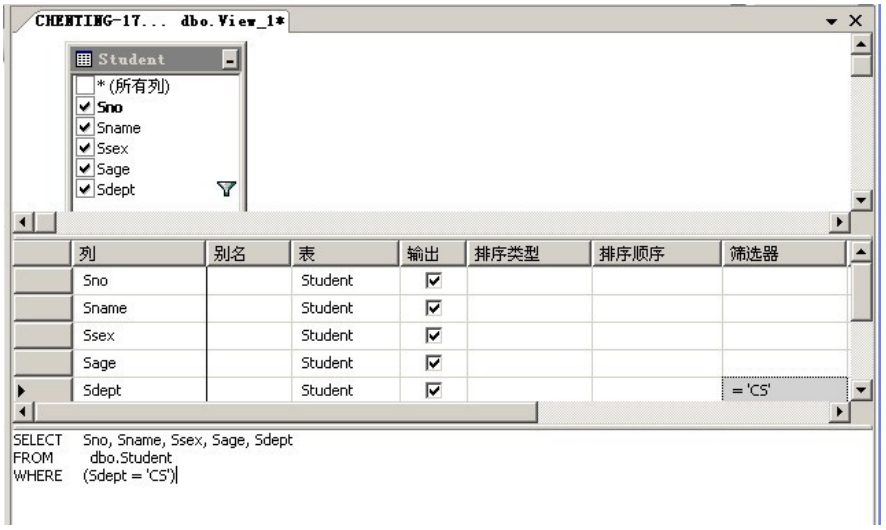


图 4-41　创建视图

设置完成后，单击【保存】按钮，出现如图 4-42 所示的【选择名称】对话框，在其中输入视图名称后，单击【确定】按钮，便完成了视图【CS_STU】的创建。

视图【CS_STU】创建成功后，其中便包含了所选择的字段及相应的数据，可以通过如下的操作查看其结构及内容。在【CS_STU】视图上单击鼠标右键，选择【选择前 1000 行】命令可以查看视图中的数据；选择【设计】命令可以查看并修改视图的结构。

图 4-42　【选择名称】对话框

2．使用 Transact-SQL 语句建立视图

使用 CREATE VIEW 语句可以创建视图，语法格式如下：

```
CREATE　VIEW　view_name[( column_list )]
[WITH　ENCRYPTION]
AS　select_statement
[WITH　CHECK　OPTION]
```

其中，各参数说明如下。

- view_name：视图名。
- column_list：视图中所包含的列名。
- WITH　ENCRYPTION：对视图进行加密。
- select_statement：利用 SELECT 语句从一张或者多张表（或视图）中返回指定的列作为新视图的列。
- WITH　CHECK　OPTION：强制在视图上所进行的修改都必须符合 select_statement 所设置的限制条件。该参数可以确保数据修改后，仍然可以通过视图看到修改的数据。

【例 4.41】　创建【BA_STU】视图，包括工商管理专业学生的学号、其选修的课程号及相对应的课程成绩。

```
CREATE　VIEW BA_STU
AS
SELECT　Student.Sno ,Cno,Grade
FROM　　Student ,SC
WHERE　　Student.Sno = SC.Sno　AND　Sdept ='BA'
```

执行命令后，可以在左边窗口的【视图】节点下看到新创建的视图【BA_STU】，如图 4-43 所示。

图 4-43　视图【BA_STU】

在上例中，如果要保证对视图的修改必须满足 Sdept 为 BA 这个条件，修改如下：

```
CREATE　VIEW　BA_STU
AS
SELECT　Student.Sno ,Cno,Grade
FROM　　Student ,SC
WHERE　　Student.Sno = SC.Sno　AND　Sdept ='BA'
WITH　CHECK　OPTION
```

视图可以从基本表中导出，如例 4-41 中的表【Student】和表【SC】，也可以从视图中导出。

【例 4.42】　为工商管理专业的学生创建他们平均成绩的视图【BA_STU_AVG】，包括学号（在视图中列名为学号）和平均成绩（在视图中列名为平均成绩）。

```
CREATE　VIEW　BA_STU_AVG（列名,平均成绩）
AS
SELECT　Sno,AVG(Grade)
FROM　　BA_STU
GROUP　 BY　Sno
```

4.3.3　查询视图

图 4-44　查询视图

视图定义完成后，就可以像查询基本表那样对视图进行查询了。

【例 4.43】　使用视图【BA_STU】查询工商管理专业的学生的学号和选修的课程号。

```
SELECT　Sno,Cno
FROM　　BA_STU
```

结果如图 4-44 所示。

视图的创建实际上简化了用户的 SQL 语句，向用户隐藏了复杂的表与表之间的连接。

4.3.4　更新视图

更新视图是指通过视图插入数据、修改数据和删除数据。对视图的更新操作通过转换，最后会成为对创建视图时所依据的基本表的更新操作。

1．插入数据（INSERT）

使用 INSERT 语句可以通过视图向基本表插入数据，语法格式如下：

```
INSERT　INTO　view_name　VALUES (column 1,column 2,column 3,…,column n)
```

【例 4.44】　向【CS_STU】视图中插入一条记录。

```
('50901010','LiuXin','M',19,'CS')
INSERT　INTO　CS_STU
VALUES('50901010','LiuXin','M',19,'CS')
```

运行结果如图 4-45 所示。

系统在执行此语句时，首先从数据字典中找到视图【CS_STU】的定义，然后将其定义和插入数据的操作结合起来，转换成等价的对基本表【Student】的插入数据的操作。我们再来验证一下在表【Student】中是不是通过视图【CS_STU】真的被插入了一行记录。

使用 SELECT 语句查询【CS_STU】依据的基本表【Student】：

```
SELECT　*　FROM　Student
```

运行结果如图 4-46 所示。

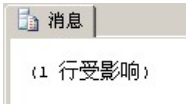


图 4-45　更新视图

结果　消息

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	18	CS
2	50901002	ZhangQiang	M	19	BA
3	50901003	HuangTing	F	18	FL
4	50901004	YeHuang	F	20	BA
5	50901005	XuYan	F	17	CS
6	50901010	LiuXin	M	19	CS

图 4-46　更新视图后的查询结果

从图 4-46 中可以看出，记录('50901010','LiuXin','M',19,'CS')已经添加到表【Student】中了。

当视图所依赖的基本表有两个或两个以上时，不能向视图中插入数据，因为修改会影响多个基本表。例如，不能向视图【BA_STU】插入数据，因为【BA_STU】依赖两个基本表:【Student】和【SC】。

【例 4.45】　向【BA_STU】视图中插入一条记录。

```
('50901010','1',88)
```

运行结果如图 4-47 所示。

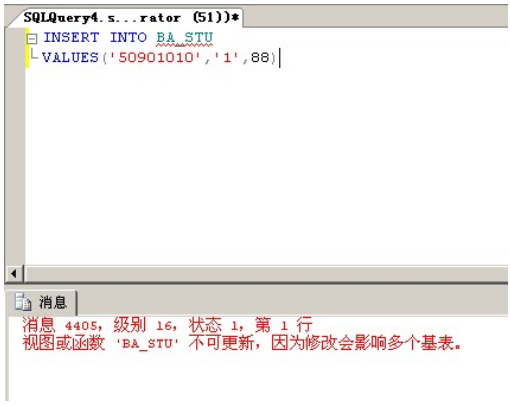


图 4-47　更新视图【BA_STU】

2．修改数据

使用 UPDATE 语句可以通过视图修改基本表的一个或多个列或者行，语法格式如下：

```
UPDATE    view_name
SET    column 1 ＝ column_value 1
          column 2 ＝ column_value 2
          ……
         column n ＝ column_value n
```

【例 4.46】　将【CS_STU】视图中所有学生的年龄减少一岁。

```
UPDATE    CS_STU
SET    Sage = Sage - 1
```

运行结果如图 4-48 所示。

再使用 SELECT 语句查询【CS_STU】依据的基本表【Student】:

```
SELECT　*　FROM　Student
```

运行结果如图 4-49 所示。

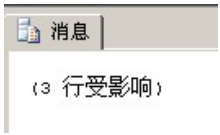


图 4-48　修改视图【CS_STU】中的数据

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	17	CS
2	50901002	ZhangQiang	M	19	BA
3	50901003	HuangTing	F	18	FL
4	50901004	YeHuang	F	20	BA
5	50901005	XuYan	F	16	CS
6	50901010	LiuXin	M	18	CS

图 4-49　查询表【Student】

将图 4-49 与图 4-46 进行比较可以得出，例 4-46 的 Transact-SQL 语句实际上是将【CS_STU】视图所依据的基本表【Student】中所有 Sdept 值为'CS'的记录的年龄字段值在原来基础上减少了一岁。

如果一个视图所依赖的基本表有两个或者两个以上，则一次修改该视图只能变动一个基本表中的数据。

【例 4.47】　将视图【BA_STU】中学号为 50901002 的学生的 3 号课程成绩修改为 85 分。

```
UPDATE BA_STU
SET Grade = 85
WHERE Sno = '50901002'　AND　Cno = '3'
```

结果如图 4-50 所示。

我们将视图【BA_STU】中更新前的数据与更新后的数据进行对比，如图 4-51 所示。

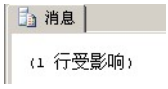
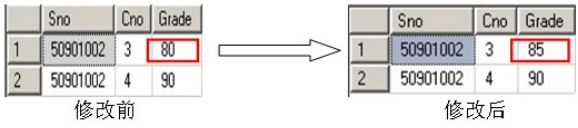


图 4-50　修改视图 BA_STU 中的数据



	Sno	Cno	Grade
1	50901002	3	80
2	50901002	4	90

修改前

	Sno	Cno	Grade
1	50901002	3	85
2	50901002	4	90

修改后

图 4-51　数据修改前后对比

3．删除数据

使用 DELETE 语句可以通过视图删除基本表的数据，语法格式如下：

```
DELETE　FROM　view_name
WHERE　search_condition
```

【例 4-48】　删除视图【CS_STU】中男同学的记录。

```
DELETE　FROM　CS_STU
WHERE　Ssex = 'M'
```

需要注意的是，对于依赖的基本表有两个或者两个以上的视图，不能使用 DELETE 语句。例如，不能通过对【BA_STU】视图执行 DELETE 语句而删除与之相关的基本表【Student】和表【CS】中的数据。

4.3.5　修改视图的定义

修改视图的定义有两种方法：一种是通过对象资源管理器窗口修改视图的定义，一种是使用 Transact-SQL 语句中的 ALTER VIEW 语句。

1．在 SQL Server Management Studio 中修改视图的定义

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【视图】→【dbo.CS_STU】，单击鼠标右键，在弹出的快捷菜单上选择【设计】命令，将出现如图 4-52 所示的窗口。

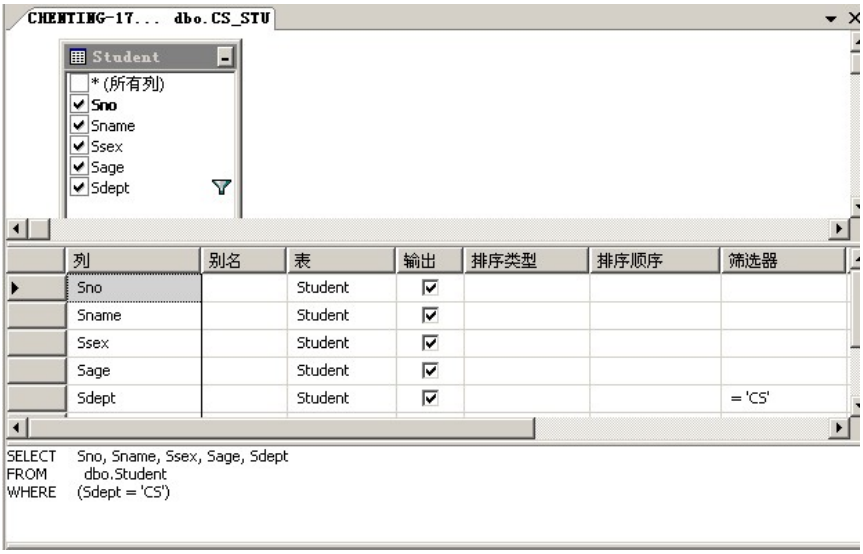


图 4-52　修改视图定义

（3）在如图 4-52 所示的窗口中对视图定义进行修改，修改完成后单击【保存】按钮即可。

2．使用 Transact-SQL 语句修改视图

使用 ALTER VIEW 语句可以修改视图，语法格式如下：

```
ALTER　VIEW　view_name
[WITH　ENCRYPTION]
AS　select_statement
[WITH　CHECK　OPTION]
```

其中，view_name、WITH ENCRYPTION、select_statement 等参数与 CREATE VIEW 语句中的含义相同。

【例 4.49】　将视图【CS_STU】修改为只包含 CS 专业学生的姓名、年龄和系别。

```
ALTER　VIEW　CS_STU
AS
SELECT　Sname,Sage,Sdept
FROM　Student
WHERE　Sdept = 'CS'
```

4.3.6　删除视图

删除视图同样有两种方法：一种是通过对象资源管理器窗口删除视图，一种是使用 Transact-SQL 语句中的 DROP VIEW 语句。

1．在对象资源管理器窗口中删除视图

以在【db_stu】数据库中删除视图【CS_STU】为例，说明在 SQL Server Management Studio 中删除视图的过程。

（1）使用【SQL Server Management Studio】连接 SQL Server 2008 服务器。

（2）在【对象资源管理器】窗口中展开连接的服务器中的【数据库】→【db_stu】→【视图】→【dbo.CS_STU】，单击鼠标右键，在弹出的快捷菜单上选择【删除】命令。

（3）单击【确定】按钮，即可完成视图的删除操作。

2．使用 Transact-SQL 语句删除视图

使用 DROP VIEW 语句可以删除视图，语法格式如下：

```
DROP　VIEW　view_name[ ,…n ]
```

其中，view_name 是要删除的视图的名称。

可以使用 DROP VIEW 语句一次删除多个视图。例如：

```
DROP　VIEW　BA_STU
```

将删除视图【BA_STU】。

4.4　游　　标

4.4.1　游标概念

我们知道，关系数据库管理系统的实质是面向集合的。在 Microsoft SQL Server 中，并没有提供一种描述表中单一记录的表达方式，除非使用 WHERE 语句来限制只有一条记录被选中。在许多应用程序中，尤其是 Transact-SQL 嵌入到的主语言（如 C 语言、VB 语言、PowerBuilder 或其他开发工具）中，通常不能把整个结果集作为一个单元来处理。因此，我们必须借助一种机制来进行面向单条记录的数据处理，这种机制就是游标（CURSOR）。

SQL Server 通过游标允许用户访问单一的记录，而并非对整个结果集进行操作（通过使用 SELECT 语句、UPDATE 语句或者 DELETE 语句进行）。用户可以对一个结果集逐行进行处理，这样可以降低系统的开销和潜在的阻隔。而且，游标把作为面向集合的数据库管理系统和面向行的程序设计紧密联系起来，使两个数据处理方式能够进行沟通。

游标具有如下的优点，正是这些优点的存在，才让游标在实际应用中发挥了重要的作用。

（1）允许应用程序对查询语句 SELECT 返回的行结果集中的每一行进行相同或者不同的操作，而不是一次对整个结果集进行相同的操作。

（2）提供基于游标位置对表中数据进行更新或删除的能力。

（3）它是面向集合的数据库管理系统（DBMS）和面向行的程序设计进行沟通的桥梁。

SQL Server 对游标的使用通常遵循的原则是：声明游标→打开游标→读取数据→关闭游标→删除游标。

4.4.2　声明游标

通常使用 DECLARE CURSOR 语句来声明一个游标，其语法格式为：

```
DECLARE  cursor_name  CURSOR
FOR  select_statement
[FOR  {READ  ONLY | UPDATE [OF  column_name[,…n]]}]
```

其中，各参数说明如下。

- cursor_name：游标的名字，它是与某个查询结果集相联系的符号名，要符合 SQL Server 标志符命名规则。
- select_statement：产生与所声明的游标相关联的结果集的 SELECT 语句。它可以是一个完整语法和语义的 SELECT 语句，但是这个 SELECT 语句必须有 FORM 子句，并且不能出现 COMPUTE、INTO 子句。
- FOR　READ　ONLY：指出所声明的游标只能读，不能修改。
- FOR　UPDATE：指出所声明的游标可以修改。
- OF　column_name[,…n]：列出可以被修改的列的名称。如果在 UPDATE 中未指出列名，则表示可以修改所有的列。

【例 4.50】　定义一个标准游标。

```
DECLARE  cursor1  CURSOR
FOR
SELECT  *
FROM  Student
WHERE  Sdept = 'CS'
```

【例 4.51】　定义一个只读游标。

```
DECLARE  cursor2  CURSOR
FOR
SELECT  *
FROM  Student
WHERE  Sdept = 'CS'
      FOR  READ  ONLY
```

【例 4.52】　定义一个可以修改的游标。

```
DECLARE  cursor3  CURSOR
FOR
SELECT  *
FROM  Student
WHERE  Sdept = 'CS'
    FOR  UPDATE
```

4.4.3　打开游标

声明游标后，要使用游标从中提取数据，就必须先打开游标，其语法格式为：

```
OPEN　cursor_name
```

其中，cursor_name 是一个已经声明的但是尚未被打开的游标的名字。

OPEN 语句可打开已被声明的游标，分析声明这个游标的 select_statement，并使结果集对于处理是可用的。

例如，OPEN cursor1 语句可打开例 4-49 中定义的游标 cursor1，并使结果集处于可用状态。

4.4.4　读取数据

游标打开以后，游标指针位于结果集的第一行之前，可以使用 FETCH 语句从结果集中读取行。SQL Server 将沿着游标结果集一行或者多行地向下移动游标指针，不断读取结果集中的数据，并修改和保存游标指针当前的位置，直到结果集中的行全部被读取。其语法格式为：

```
FETCH
[[NEXT | PRIOR | FIRST | LAST | ABSOLUTE n] FROM]　cursor_name
[INTO　fetch_target_list]
```

其中，各参数说明如下。

- cursor_name：表示一个已经声明并且打开的游标的名字，将从此游标中读取数据。
- NEXT | PRIOR | FIRST | LAST | ABSOLUTE n：表示移动游标指针的方法。其中，NEXT 表示下移一条记录，读取当前记录的下一条记录，并且使其置为当前记录；PRIOR 表示上移一条记录，读取当前记录的上一条记录，并且使其置为当前记录；FIRST 表示读取结果集中的第一条记录，并且使其置为当前记录；LAST 表示读取结果集中的最后一条记录，并且使其置为当前记录；ABSOLUTE　n 表示读取第 n 条绝对记录，n 必须为整形常量。如果不指定，默认情况是 NEXT，即向下移动一条记录。
- I NTO　fetch_target_list：表示将读取的游标数据存放到指定的变量清单中。这个变量清单中变量的个数、数据类型及顺序必须与声明游标的 select_statement 的 select_list 中列出的列清单相匹配。

【例 4.53】　假设游标 cursor1 已经打开，从该游标中读取数据。

```
FETCH　NEXT　FROM　cursor1
```

结果如图 4-53 所示。

	Sno	Sname	Ssex	Sage	Sdept
1	50901001	WangYu	M	17	CS

图 4-53　从游标 cursor1 中读取数据

从图 4-53 中可以看出，默认情况下，每次执行 FETCH 语句，只返回结果集中的一行。

游标指针确定结果集中哪一条记录可以被读取；如果游标方式为 FOR UPDATE，则游标指针确定结果集中哪一条记录可以被更新。

FETCH 语句的执行状态保存在全局变量@@FETCH_STATUS 中，如表 4-12 所示。

表 4-12　FETCH 语句的状态信息

值	说　明
0	表示成功执行 FETCH 语句
−1	表示 FETCH 语句有错误，或者当前游标指针已经指向结果集中的最后一条记录
−2	表示被读取的记录不存在（已被删除）

例如，接着例 4-52 继续执行如下语句：

```
FETCH　NEXT　FROM　cursor1
SELECT　'FETCH 执行情况' = @@FETCH_STATUS
```

结果如图 4-54 所示。

图 4-54　查看 FETCH 执行情况

4.4.5　关闭游标

暂时关闭游标可以使用 CLOSE 语句，它并不会改变游标的声明，只是停止处理声明游标的那个 select_statement，等到需要使用的时候，可以再次用 OPEN 语句打开游标，SQL Server 会用该游标的定义重新创建这个游标的一个结果集。关闭游标的语法格式为：

```
CLOSE　cursor_name
```

其中，cursor_name 是一个已经被打开的游标的名字。

例如，CLOSE cursor1 将关闭游标 cursor1。

4.4.6　删除游标

使用 CLOSE 语句关闭游标以后，游标的定义仍在，可以使用 OPEN 语句再次打开它。如果确认游标不再需要，就应该释放分配给此游标的资源，即从内存中删除游标，包括该游标的名字。此时，如果还想使用，必须重新声明。

删除游标的语法格式为：

```
DEALLOCATE　cursor_name
```

其中，cursor_name 表示已打开或已关闭的游标的名字。如果删除一个已打开未关闭的游标，SQL Server 2008 会先自动关闭这个游标，然后再删除它。

例如，DEALLOCATE cursor1 将删除游标 cursor1。

习　题

1．试说明 SELECT 语句的作用。

2．试说明 SELECT 语句的 FROM、WHERE、GROUP BY 及 ORDER BY 子句的作用。

3. SQL 语句中，条件年龄 BETWEEN 17 AND 19 表示年龄在 17 和 19 之间，并且（　　）。

A．包括 17 岁和 19 岁　　B．包括 17 岁但不包括 19 岁

C．不包括 17 岁和 19 岁　　D．包括 19 岁但不包括 17 岁

4．SQL Server 2008 中表查询的命令是（　　）。

A．USE　　B．SELECT　　C．UPDATE　　D．DROP

5．在 SELECT 语句的 WHERE 子句的条件表达式中，可以匹配 0 个到多个字符的通配符是（　　）。

A．*　　B．%　　C．_　　D．?

6．SQL 语句中，下列涉及空值的操作，不正确的是（　　）

A．AGE IS NULL　　B．AGE ＝ NULL

C．AGE is not NULL　　D．NOT (AGE IS NULL)

7．在 SQL 的查询语句中，GROUP BY 选项可实现对结果表的（　　）功能。

A．分组统计　　B．求和　　C．查找　　D．排序

8．在 SELECT 语句中使用 * ，表示（　　）。

A．选择全部属性　　B．选择任意属性

C．选择全部元组　　D．选择主码

9．在 SELECT 语句中使用 GROUP BY Sname 时，Sname 必须（　　）

A．在 WHERE 中出现　　B．在 FROM 中出现

C．在 SELECT 中出现　　D．在 HAVING 中出现

10．写出 Transact-SQL 语句，对表【Student】、【Course】和【SC】进行如下操作。

（1）查询考试成绩不及格的学生的学号。

（2）查询课程表的详细信息，查询结果按照学分降序排列，相同学分的按照课时数升序排列。

（3）查询姓名中含有“i”的学生的详细信息。

（4）查询与 XuYan 不在同一个系学习的学生的名单。

（5）查询选修 BIT 课程的男生姓名。

（6）查询至少有 4 个学生选修的课程名称（要求分别用连接查询和嵌套查询完成）。

（7）查询选修了两门以上课程的学生的姓名和平均成绩。

11．举例说明游标的使用步骤。

第 5 章　T-SQL 语言

对于 SQL Server 数据库的学习来说，T-SQL 程序设计是其中重要的环节之一，它对以后的程序开发有着直接的决定性因素。本章从 T-SQL 程序设计中的数据类型及常量、变量的定义入手，重点讨论了流程控制语句及函数的定义和使用方法。

5.1　常量、变量与数据类型

5.1.1　常量

在程序的运行过程中，其值不能被改变的数据称为常量。常量的格式取决于它所表示的值的数据类型。

1．字符串常量

字符串常量用一对单引号（' '）定界，包含字母和数字字符（a～z、A～Z 和 0～9），以及特殊字符，如感叹号（!）、at 符（@）和数字号（#）等。以下是字符串的示例：

```
'abcdefg'
'Progress  B  has  completed  60%. '
'10 +40=50 '
```

如果字符串中本身又有单引号字符，可以使用两个单引号表示嵌入的单引号，例如：

```
'Apostrophe  is  displayed  like  this:can ' 't '
```

2．Unicode 字符串

Unicode 字符串的格式与普通字符串的格式类似，但它前面有一个 N 标志符（N 代表 SQL-92 标准中的区域语言）。需要注意的是，N 前缀必须是大写字母。例如，'China'是字符串常量，而 N'China'则是 Unicode 常量。

除了在使用方式上 Unicode 字符串需要加上 N 标志符以外，在字符数据的存储上，普通字符串每个字符使用 1 个字节存储，而 Unicode 字符串则要使用 2 个字节存储。

3．二进制字符串常量

二进制字符串常量是指以 0x 开头，后面由若干十六进制数字组成的常量。二进制字符串常量不使用引号括起。例如：

```
0x54321D0C
0xBE
0x34Af
```

4．bit 常量

bit 常量使用数字 0 和 1 表示，并且不括在引号中。如果使用一个大于 1 的数字，则该数字转换为 1。

5．datetime 常量

datetime 常量使用特定格式的字符日期值来表示，并用单引号括起来。例如：

```
'December 24,2009'
'24 December,2009'
'091224'
'12/25/09'
'15:15:15'
'03:15 PM'
```

6．整型常量

整型常量由没有用引号括起来并且不包含小数点的数字字符串来表示。例如：

```
2009
12
+3456
−245643
```

7．decimal 常量

decimal 常量由没有用引号括起来并且包含小数点的数字字符串来表示。例如：

```
2009.10
2.56
+3.1415926
−345678.2314
```

8．float 和 real 常量

float 和 real 常量主要采用科学记数法来表示。例如：

```
202.2E4
2.3E-5
+123E-4
−34E2
```

9．money 常量

money 常量是以“$”作为前缀的一个整型或实型常量数据（包括 decimal 常量、float 常量和 real 常量）。例如：

```
$543
$543.21
+$46.78
−$46.78
```

10．uniqueidentifier 常量

uniqueidentifier 常量是指表示 GUID（全局唯一标志符）的字符串。可以使用字符或者二进制字符串格式指定。它由 32 个任意字符组成。例如，以下就是有效的 GUID：

```
        0xff19966f868b11d0b42d00c04fc964ff
'6F9619FF-8A86-D011-B42D-00004FC964FF'
```

5.1.2　数据类型

每一个属性列的值都必须来自于同一个数据类型，我们可以将 Transact-SQL 中使用的数据类型归纳为以下几类：

- Numeric 数据类型；
- Character 数据类型；
- Temporal(date and/or time) 数据类型；
- Miscellaneous 数据类型。

下面，我们对以上的数据类型逐一介绍。

1．Numeric 数据类型

Numeric 数据类型用来表示数据。如表 5-1 所示列出了所有的 Numeric 数据类型。

表 5-1　Numeric 数据类型

数据类型	存放字节数		范　围	备　注
INTEGER	4		−2 147 483 648～2 147 483 647	
SMALLINT	2		−32 768～32 767	
TINYINT	1		0～255	
BIGINT	8		-2^{63}～$2^{63}-1$	
DECIMAL(p,[s])			$-10^{38}+1$～$10^{38}-1$	p（精度）表示最多可以存储的十进制数字的总位数，包括小数点左边和右边的位数。P 的取值范围是 1~38，默认值是 38。 s（小数位数）表示小数点右边可以存储的十进制数字的最大位数，小数位数必须是 0 到 p 之间的值。仅在指定精度后才可以指定小数位数，默认的小数位数是 0
NUMERIC(p,[s])				功能上等价于 DECIMAL(p,[s])
REAL	4		−3.40E＋38～−1.18E−38、0 以及 1.18E−38～3.40E＋38	
FLOAT[(p)]	[1,24]	[25,53]		P 是指用于存储 FLOAT 数值尾数的位数(以科学记数法表示)，p 的取值范围是 1~53，默认值是 53
	4	8		
MONEY	8		−922 337 203 685 477.580 8～922 337 203 685 477.580 7	
SMALLMONEY	4		−214 748.364 8～214 748.364 8	

2．Character 数据类型

Character 数据类型分为字符串和 Unicode 字符串。如表 5-2 所示列出了所有的 Character 数据类型。

表 5-2　Character 数据类型

数据类型	解　　释
CHAR[(n)]	表示长度为 n 个字节的固定长度的非 Unicode 字符数据，n 的取值范围是 1～8 000，存储大小是 n 个字节
VARCHAR[(n)]	表示长度为 n 个字节的可变长度的非 Unicode 字符数据，n 的取值范围是 1～8 000，存储大小是输入数据的实际长度加 2 个字节
NCHAR[(n)]	表示长度为 n 个字符的固定长度的 Unicode 字符数据，n 的取值范围是 1～4 000，存储大小是 n 个字节的两倍
NVARCHAR[(n)]	表示长度为 n 个字符的可变长度的 Unicode 字符数据，n 的取值范围是 1～4 000，存储大小是 n 个字节的两倍加 2 个字节

3．Temporal（date and/or time）数据类型

Transact-SQL 中支持的 Temporal 数据类型有 DATETIME、SMALLDATETIME、DATE、TIME、DATETIME2 和 DATETIMEOFFSET。

DATETIME 和 SMALLDATETIME 可以存储各种文本形式的日期和时间数据。两者的区别就是各自能够存储的值的范围。DATETIME 可以存储从 1753 年 1 月 1 日至 9999 年 12 月 31 日之间的值，并且还能精确到 3.33 毫秒。SMALLDATETIME 可以存储从 1900 年 1 月 1 日至 2079 年 6 月 6 日之间的值，并且只能精确到 1 分钟。对于存储空间来说，DATETIME 需要 8 个字节，而 SMALLDATETIME 只需要 4 个字节。

如果只想存储日期部分或者时间部分，那么 DATETIME 和 SMALLDATETIME 使用起来会非常不方便。所以，SQL Server 2008 做了一些改进，增加了新的数据类型 DATE 和 TIME，它们分别存储 DATETIME 数据类型的日期和时间部分。DATE 需要 3 个字节的存储空间，可以存储从 0001 年 1 月 1 日至 9999 年 12 月 31 日的日期值。TIME 需要 3～5 个字节的存储空间，可以精确到 100 纳秒。

DATETIME2 也是 SQL Server 2008 中新增加的数据类型，可以将其视为 DATETIME 数据类型的扩展，它扩展了可以接受日期的范围，以及在 DATETIME 时间部分添加附加精度。DATETIME2 可以存储从 0001 年 1 月 1 日至 9999 年 1 月 1 日之间的值，其时间部分的精确度也是 100 纳秒。

DATETIMEOFFSET 也是 SQL Server 2008 中新增加的一种数据类型，用来定义一个日期和时间的组合。其中时间部分是以 24 小时制显示的，可以精确到 100 纳秒，并且带有时区提示。在 DATETIMEOFFSET 数据类型中加入了时区偏移量部分，表示为[+|–] HH:MM。HH 的取值范围是从 00～14 的 2 位数，表示时区偏移量的附加小时数。MM 的取值范围是从 00～59 的 2 位数，表示时区偏移量的附加分钟数。

4．Miscellaneous 数据类型

Transact-SQL 中支持的某些数据类型并不属于先前介绍过的数据类型中的任何一种，我们将它们归结为 Miscellaneous 数据类型。

1）BINARY 和 VARBINARY 数据类型

BINARY 和 VARBINARY 数据类型分别用来表示固定长度和可变长度的 BINARY 数据类型，如表 5-3 所示。

表 5-3　BINARY 和 VARBINARY 数据类型

数 据 类 型	解　　释
BINARY[(n)]	表示长度为 n 字节的固定长度的二进制数据，n 的取值范围是 1～8 000，存储大小是 n 个字节
VARBINARY[(n)]	表示可变长度的二进制数据，n 的取值范围是 1～8 000，存储大小是输入数据的实际长度加 2 个字节

2）BIT 数据类型

BIT 表示可以取值为 0、1 或者 NULL 的整数数据类型。

3）LARGE OBJECTS 数据类型

LARGE OBJECTS（LOBS）可以存储最大为 2GB 的数据对象，常常被用来存储大容量的文本数据、加载模块和音频/视频文件。Transact-SQL 支持两种不同的指定和访问 LOBS 的方式，一种是使用 VARCHAR（MAX）、NVARCHAR（MAX）和 VARBINARY（MAX）数据类型，另一种是使用 TEXT/IMAGE 数据类型。

4）UNIQUEIDENTIFIER 数据类型

UNIQUEIDENTIFIER 数据类型可以存储 16 字节的二进制值，其作用与 GUID（全局唯一标志符）一样。

5）SQL_VARIANT 数据类型

SQL_VARIANT 数据类型用于存储 SQL Server 2008 支持的各种数据类型（不包括 TIMESTAMP）的值。

6）HIERARCHYID 数据类型

HIERARCHYID 数据类型是一种长度可变的系统数据类型，它是 SQL Server 2008 新增加的一种数据类型，主要解决拥有树形层次关系的表格问题。

7）TIMESTAMP 数据类型

TIMESTAMP 数据类型是一种自动生成的二进制数字，并且这些数字在数据库中是唯一的，一般用来给表行加版本戳。

5.1.3　变量

1．变量的定义

我们经常需要保存或使用一个不是直接来自于某个属性列的值，也可能该值是从一行或一列中提取的，而用户希望将它应用于查询中的不同位置，这时就需要使用变量。

2．变量的赋值方法

在 Transact-SQL 语句中变量有两种形式：一种是用户可以自定义的变量，称为局部变量；一种是系统提供的全局变量。

1）局部变量

局部变量是一个有特定数据类型的对象，并且作用范围仅局限于程序内部。局部变量被引用时，要在其名称前加上标志符@，而且必须先用 DECLARE 命令定义后才可以使用。其语法格式如下：

```
DECLARE　@local_variable　datatype [＝ value][,@local_variable　datatype [= value]　…]
```

其中，各参数说明如下。

- @local_variable：变量名称，需要注意的是，变量名不允许是 Transact-SQL 的保留字，不允许有空格出现在变量名内，而且局部变量的名称与后面讲到的全局变量的名称不能重名，否则会在应用程序中出现不可预测的后果。
- datatype：变量的数据类型。
- value：变量的值。

在 Transact-SQL 语句中，也可以先定义局部变量，再使用 SELECT 或 SET 语句，其语法格式如下：

```
SELECT  @local_variable = value
SET  @local_variable = value
```

其中 SET 语句每次只能对一个局部变量赋值，SELECT 语句可以同时给一个或多个局部变量赋值。

2）全局变量

全局变量是 SQL Server 系统提供并赋值的变量，用户不能定义全局变量，也不能使用 SET 语句修改全局变量的值。引用全局变量时，必须以标志符@@引导。

当我们在查询窗口中输入@@时，旁边会自动出现提示，通过列表显示所有的全局变量，如图 5-1 所示，用户可以根据自己的需要选择。

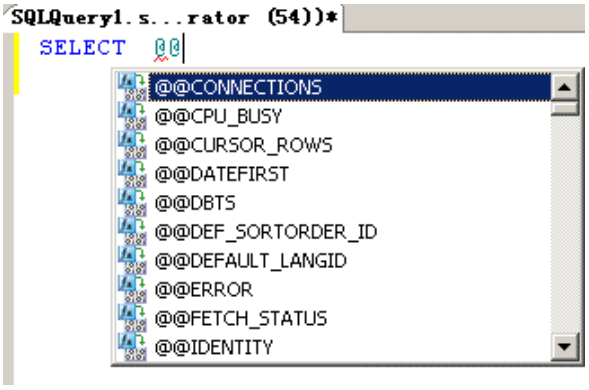


图 5-1　全局变量列表

3．变量使用举例

【例 5.1】　查询平均成绩大于 80 分的学生的学号。

```
DECLARE  @score  int
SET  @score = 80
SELECT  Sno
FROM  SC
GROUP  BY  Sno
HAVING  AVG(Grade) > @score
```

运行结果如图 5-2 所示。

等价于：

```
DECLARE　@score　int = 80
SELECT　Sno
FROM　SC
GROUP　BY　Sno
HAVING　AVG ( Grade ) > @score
```

	Sno
1	50901002

图 5-2　使用局部变量运行结果一

【例 5.2】　查询学号为 50901001 的学生的姓名及年龄。

```
DECLARE　@name　char(40),@age　　int
SELECT　@name = Sname , @age = Sage
FROM　Student
WHERE　Sno =　'50901001'
SELECT　　@name , @age
```

运行结果如图 5-3 所示。

【例 5.3】　查询当前使用的语言。

```
SELECT　@@LANGUAGE
```

运行结果如图 5-4 所示。

	(无列名)	(无列名)
1	WangYu	18

图 5-3　使用局部变量运行结果二

	(无列名)
1	简体中文

图 5-4　使用全局变量运行结果

5.2　运算符与表达式

SQL Server 2008 提供了算术运算符、赋值运算符、比较运算符、位运算符、一元运算符、字符串连接运算符及复合运算符。

1．算术运算符

算术运算符用于在两个表达式上执行算术运算，这两个表达式可以是任何数字数据类型。

算术运算符包括：加（＋）、减（－）、乘（*）、除（/）和取模（%）。

【例 5.4】　求学生的出生年份。

```
DECLARE　@date　datetime
SET　@date = getdate()
SELECT　@date – Sage　AS　出生年份
FROM　Student
```

2．赋值运算符

赋值运算符（=）可以将数据值指派给特定的对象，如给局部变量赋值的 SET 语句和 SELECT 语句中使用的“=”。

3．比较运算符

比较运算符又称为关系运算符，可对其左右两边表达式的大小关系进行判断。比较运算符的计算结果为布尔数据类型，如果关系成立，则返回 TRUE；如果关系不成立，则返回 FALSE。

除了 text、ntext 和 image 数据类型的表达式外，比较运算符可以应用于所有的表达式。

1）等于（=）

对于非空的参数，如果左边的参数等于右边的参数，则返回 TRUE，否则返回 FALSE。如果任一参数的计算结果等于空值或两个参数的计算结果都等于空值，此运算符返回空值。除非进行了 0=NULL 的比较，在这种情况下，布尔值中将包含 TRUE。

2）不等于（<>、!=）

对于非空的参数，如果左边的参数不等于右边的参数，则返回 TURE，否则返回 FALSE。如果其中一个参数的计算结果等于空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

3）大于（>）

对于非空的参数，如果左边的参数值大于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

4）大于或等于（>=）

对于非空的参数，如果左边的参数值大于或等于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

5）小于（<）

对于非空的参数，如果左边的参数值小于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

6）小于或等于（<=）

对于非空的参数，如果左边的参数值小于或等于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

7）不大于（!>）

对于非空的参数，如果左边的参数值小于或等于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

8）不小于（!<）

对于非空的参数，如果左边的参数值大于或等于右边的参数值，则返回 TRUE，否则返回 FALSE。如果其中一个参数的计算结果为空值或这两个参数的计算结果都等于空值，则该运算符返回空值。

4．位运算符

位运算符在两个表达式之间进行位操作，这两个表达式可以为整数数据类型中的任何数据类型。

1）AND（&）

双目运算符，对两个数值表达式进行逻辑与运算，运算规则如表 5-4 所示。

表 5-4　AND 运算规则

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

2）IS

双目运算符，对两个对象表达式进行逻辑比较，运算规则如表 5-5 所示。

表 5-5　IS 运算规则

X	Y	X IS Y
1	1	1
1	0	0
0	1	0
0	0	1

3）NOT（～）

单目运算符，对数值表达式进行逻辑非运算，运算规则如表 5-6 所示。

表 5-6　NOT 运算规则

X	NOT X
1	0
0	1

4）OR（|）

双目运算符，对两个数值表达式进行逻辑或运算，运算规则如表 5-7 所示。

表 5-7　OR 运算规则

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

5）XOR（^）

双目运算符，对两个数值表达式进行逻辑异或运算，运算规则如表 5-8 所示。

表 5-8　XOR 运算规则

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

6）ALL

如果一组的比较都为 TRUE，那么运算结果就为 TRUE。

>ALL 表示大于每一个值，即大于最大值。例如，>ALL(2,3,4)表示大于 4，因此，使用>ALL 的子查询也可以用 MAX 集函数实现。

<ALL 表示小于每一个值，即小于最小值。例如，<ALL(2,3,4)表示小于 2，因此，使用<ALL 的子查询也可以用 MIN 集函数实现。

<>ALL 与 NOT IN 等效。

7）ANY

如果一组的比较中任何一个为 TRUE，那么运算结果就为 TRUE。

>ANY 表示至少大于一个值，即大于最小值。例如，>ANY(2,3,4)表示大于 2，因此，使用>ANY 的子查询也可以用 MIN 集函数实现。

<ANY 表示至少小于一个值，即小于最大值。例如，<ANY(2,3,4)表示小于 4，因此，使用<ANY 的子查询也可以用 MAX 集函数实现。

=ANY 与 NOT IN 等效。

8）BETWEEN

如果操作数在指定的范围内，那么运算结果就为 TRUE。

9）EXISTS

如果子查询中包含一些行，那么运算结果就为 TRUE。

10）IN

如果操作数等于表达式列表中的一个，那么运算结果就为 TRUE。

11）LIKE

如果操作数与一种模式相匹配，那么运算结果就为 TRUE。

12）SOME

如果在一组比较中，有些值为 TRUE，那么运算结果就为 TRUE。

5．一元运算符

一元运算符只对一个表达式进行操作。

1）正（+）

正运算符用于返回数值表达式的正值，可以用于 NUMERIC 数据类型类别中的任一数据类型的任意表达式。

2）负（-）

负运算符用于返回数值表达式的负值，可以用于 NUMERIC 数据类型类别中的任一数据类型的任意表达式。

3）按位取反（～）

按位取反运算符可以用于整数数据类型类别中的任一数据类型的任意表达式。例如，设 x

的值为 15（0000 0000 0000 1111），计算～x 的值为 1111 1111 1111 0000。

6．字符串连接运算符

字符串连接运算符（+）可以将两个或多个字符串合并或串联成一个字符串，也可以串联二进制字符串。

例如，SELECT'好好学习' + '天天向上'，其结果为'好好学习天天向上'。

7．复合运算符

SQL Server 2008 中新增了如表 5-9 所示的复合运算符，它们可以执行操作并且将操作结果的值赋给变量。

表 5-9　复合运算符

运　算　符	操　　作
+=	将原始值加上一定的量，并将原始值设置为结果
-=	将原始值减去一定的量，并将原始值设置为结果
*=	将原始值乘以一定的量，并将原始值设置为结果
/=	将原始值除以一定的量，并将原始值设置为结果
%=	将原始值加上一定的量，并将原始值设置为余数
&=	对原始值执行位与运算，并将原始值设置为结果
^=	对原始值执行位异或运算，并将原始值设置为结果
|=	对原始值执行位或运算，并将原始值设置为结果

【例5.5】　复合运算符的应用。

```
DECLARE   @x1 = 18
SET   @x1 *= 2
SELECT   @x1                --返回值 36
```

8．运算符的优先级顺序

当一个复杂的表达式有多个运算符时，运算符优先级决定执行运算的先后次序。执行的顺序可能严重地影响所得到的值。

运算符的优先级如表 5-10 所示。在一个表达式中按照先高（优先级数字小）后低（优先级数字大）的顺序进行运算。

表 5-10　运算符优先级顺序表

优先级级别	运　算　符
1	～（位非）
2	*（乘）、/（除）、%（取模）
3	+（正）、－（负）、+（加）、－（减）、+（连接）、&（位与）、^（位异或）、|（位或）
4	=、>、<、>=、<=、<>、!=、!<、!>（比较运算符）
5	NOT
6	AND
7	ALL、ANY、BETWEEN、IN、LIKE、OR、SOME
8	=（赋值）

（1）当一个表达式中的两个运算符有相同的优先级的时候，将按照它们在表达式中的位置对其从左到右进行求值。

（2）在表达式中可以使用一对圆括号改变所定义的运算符的优先级顺序。先对括号内的表达式求值，然后对括号外的运算符进行运算时使用该值。

（3）如果表达式有嵌套的括号，则首先对嵌套最深的表达式进行运算。

5.3　流程控制语句

流程控制语句可以用来改变计算机的执行流程，以满足程序设计的需要。它常常用于控制 Transact-SQL 语句、语句块或存储过程的执行流程。

5.3.1　IF…ELSE 语句

利用 IF…ELSE 语句可以对给定的条件进行判定，并根据判定的结果（真或假）分别执行不同的 Transact-SQL 语句。其语法格式为：

```
IF  Boolean_expression
        {sql_statement | statement_block}
[ ELSE
        {sql_statement | statement_block} ]
```

参数说明如下。

- Boolean_expression：逻辑表达式。如果逻辑表达式中含有 SELECT 语句，则必须用括号将 SELECT 语句括起来，运算结果为 TRUE 或 FALSE。
- { sql_statement | statement_block }：任何 Transact-SQL 语句或用语句块定义的语句分组。除非使用语句块，否则 IF 或 ELSE 条件只能影响一个 Transact-SQL 语句的性能。如果要定义语句块，则要使用关键字 BEGIN 和 END。

根据上述语法格式，可以看出 ELSE 部分是可以根据实际需要省略或保留的。所以，我们将 IF…ELSE 语句细分为下面的两种形式。

```
IF  逻辑表达式
        A
```

当逻辑表达式的值为 TRUE 时，执行 A，然后执行 IF 语句的下一条语句；当逻辑表达式的值为 FALSE 时，直接跳过 A，执行 IF 语句的下一条语句。

```
IF  逻辑表达式
        A
ELSE
        B
```

当逻辑表达式的值为 TRUE 时，执行 A，然后执行 IF 语句的下一条语句；当逻辑表达式的值为 FALSE 时，执行 B，然后执行 IF 语句的下一条语句。

【例 5.6】 如果课程名为 BIT 的平均成绩高于 85 分，则显示“BIT 课程成绩为中等水平”，否则显示“BIT 课程成绩未达到中等水平”。

```
DECLARE   @text1, @text2   char(20)
SET    @text1 = 'BIT 课程成绩为中等水平'
SET    @text2 = 'BIT 课程成绩未达到中等水平'
IF( SELECT   AVG(Grade)
        FROM   Course,SC
        WHERE   Course.Cno = SC.Cno
AND Course.Cname = 'BIT' ) > 85
SELECT   @text1
ELSE
     SELECT   @text2
```

5.3.2　无条件转移（GOTO）语句

GOTO 语句和语句标号相结合，表示无条件地转移到语句标号指定的位置，然后再继续往下执行。语句标号必须以“:”结尾，在 GOTO 命令行，语句标号后不必加“:”。其语法格式为：

```
Define   the   label
label :
Alter   the   execution
GOTO   label
```

参数说明如下。

● label：如果 GOTO 语句指向该标签，则该标签为处理的起点。标签必须符合标志符规则。

【例 5.7】　利用 GOTO 语句，计算 1+2+3+4+5+…+99+100 的值。

```
DECLARE   @sum   int,   @i   int
SET   @sum = 0
SET   @i = 1
s:
       IF(@i <= 100)
          BEGIN
          SET   @sum = @sum + @i
          SET   @i = @i + 1
          GOTO   s
          END
SELECT   @sum
```

5.3.3　WHILE 语句

WHILE 语句用来设置重复执行 SQL 语句或语句块的条件，只要执行的条件为真，就重复执行语句。其语法格式为：

```
WHILE   Boolean_expression
```

```
{sql_sta ement | statement_block | BREAK | CONTINUE}
```

参数说明如下。

- Boolean_expression：逻辑表达式。如果逻辑表达式中含有 SELECT 语句，则必须用括号将 SELECT 语句括起来，运算结果为 TRUE 或 FALSE。
- { sql_statement | statement_block }：任何 Transact-SQL 语句或用语句块定义的语句分组。如果要定义语句块，则要使用关键字 BEGIN 和 END。
- BREAK：导致从最内层的 WHILE 循环中退出，将执行出现在 END 关键字（循环结束的标记）后面的任何语句。
- CONTINUE：结束本次循环，使 WHILE 循环重新开始执行，忽略 CONTINUE 关键字后面的任何语句。

例如：

```
WHILE   逻辑表达式 1
BEGIN
        A
        IF   WHILE   逻辑表达式 2
        CONTINUE
        B
    END
```

当逻辑表达式 1 为 TURE 时，执行 A，然后判断逻辑表达式 2 是否为真，若为真则跳过 B，执行 WHILE 语句，否则执行 B。

5.3.4　RETURN 语句

RETURN 语句用于从批处理、存储过程或语句块中无条件退出。RETURN 语句可以返回一个整数给调用它的过程或应用程序。除非另外说明，否则返回值 0 表示成功返回，非 0 表示失败。当用于存储过程时，RETURN 语句不能返回空值。如果某个过程试图返回空值，则将生成警告信息并返回值 0。其语法格式如下：

```
RETURN   [ integer_expression ]
```

参数说明如下。

- integer_expression：返回的整数值。存储过程可向执行调用的过程或应用程序返回一个整数值。

5.3.5　WAITFOR 语句

WAITFOR 语句用于在达到指定的时间或时间间隔之前，或者指定语句至少修改或返回一行之前，阻止执行批处理、存储过程或事务。其语法格式为：

```
WAITFOR{ DELAY 'time_to_pass' | TIME 'time_to_execute'}
```

参数说明如下。

- DELAY：可以继续执行批处理、存储过程或事务之前必须经过的指定时段，最长为 24 小时。

- 'time_to_pass'：等待的时段。可以使用 DATETIME 数据可接受的格式之一指定 time_to_pass，也可以将其指定为局部变量，但不能指定日期，因此，不允许指定 DATETIME 值的日期部分。
- TIME：指定运行批处理、存储过程或事务的时间。
- 'time_to_execute'：WAITFOR 语句完成的时间。可以使用 DATETIME 数据可接受的格式之一指定 time_to_execute，也可以将其指定为局部变量。不能指定日期，因此，不允许指定 DATETIME 值的日期部分。

【例 5.8】　等待 4 小时 4 分零 4 秒后再执行 SELECT 语句。

```
WAITFOR  DELAY  '04:04:04'
SELECT  *  FROM  Student
```

5.4　系统内置函数

函数是指一组编译好的 Transact-SQL 语句，它们可以带有一个或多个参数，也可以不带参数。函数执行的结果为返回一个数值、数值集合，或者没有返回值仅仅是执行一些操作。

SQL Server 2008 支持两种函数类型：系统内置函数和用户定义函数。本节中将重点介绍系统内置函数，有关用户定义函数的内容请参阅 5.5 节。

5.4.1　系统内置函数介绍

系统内置函数是一组预定义的函数，它是 Transact-SQL 语言的一部分，按照定义的方式运行并且不能修改。Transact-SQL 语言提供的函数包括聚合函数和纯量函数。

聚合函数适用于列中的一组数据值，返回一个标量值（单一值）。Transact-SQL 中主要支持的聚合函数有统计型聚合函数、便捷型聚合函数、用户自定义聚合函数和分析型聚合函数。

纯量函数主要针对单一值或者列表值，与聚合函数不同的是，纯量函数主要是针对多行中的数据。我们可以将纯量函数归纳为数学函数、字符串函数、日期函数、系统函数和元数据函数几类。

5.4.2　常用系统内置函数

1．数学函数

数学函数可对数字表达式进行数学运算并返回运算结果。

1）ABS

［格式］　ABS(n)

［功能］　返回数字表达式 n 的绝对值（正值）。

［示例］　SELECT　ABS(3.14)=3.14

　　SELECT　ABS(−3.14)=3.14

2）ASIN

［格式］　ASIN(n.n)

［功能］　返回正弦值为浮点表达式 n.n 的弧度角，其取值范围为−1～1。

［示例］　SELECT　ASIN(1)=−1.5707963267949

SELECT　ASIN(−1)=−1.5707963267949

3）ACOS

［格式］　ACOS(n.n)

［功能］　返回余弦值为浮点表达式 n.n 的弧度角，其取值范围为−1～1。

［示例］　SELECT　ACOS(0.1)=1.47062890563334

SELECT　ACOS(−0.1)=−1.47062890563334

4）ATAN

［格式］　ATAN(n.n)

［功能］　返回正切值为浮点表达式 n.n 的弧度角。

［示例］　SELECT　ATAN(3.14)=1.26248066459947

SELECT　ATAN(−3.14)=−1.26248066459947

5）ATN2

［格式］　ATN2(x,y)

［功能］　返回由 X 轴到点（x,y）的弧度角。

［示例］　SELECT　ATN2(0,1)=0

SELECT　ATN2(1,0)=1.5707963267949

6）SIN

［格式］　SIN(n.n)

［功能］　返回数字表达式 n.n 的正弦值。

［示例］　SELECT　SIN(3.14)=0.00159265291648683

SELECT　SIN(2)=0.909297426825682

7）COS

［格式］　COS(n.n)

［功能］　返回浮点表达式 n.n 的余弦值。

［示例］　SELECT　COS(3.14)=−0.99999873172754

SELECT　COS(0.5)=0.877582561890373

8）TAN

［格式］　TAN(n.n)

［功能］　返回浮点表达式 n.n 的正切值。

［示例］　SELECT　TAN(0)=0

SELECT　TAN(1)=1.5574077246549

9）COT

［格式］　COT(n.n)

［功能］　返回浮点表达式 n.n 的余切值。

［示例］　SELECT　COT(3.14)=−627.882397586913

SELECT　COT(1)=0.642092615934331

10）CEILING

［格式］　CEILING(n)

［功能］　返回大于或等于数字表达式 n 的最小整数。

［示例］　SELECT　CEILING(3.14)=4

SELECT　CEILING(−3.14)=−3

11）DEGREES

［格式］　DEGREES(n)

［功能］　将指定的弧度 n 转换为角度。

［示例］　SELECT　DEGREES(PI()/3)=60.0

SELECT　DEGREES(0.75)=42.971834634811742000

12）EXP

［格式］　EXP(n)

［功能］　返回 e 的 n 次方。

［示例］　SELECT　EXP(1)=2.71828182845905

13）FLOOR

［格式］　FLOOR(n)

［功能］　返回小于或等于数字表达式 n 的最大整数。

［示例］　SELECT　FLOOR(3.14)=3

SELECT　FLOOR(−3.14)=−4

14）LOG

［格式］　LOG(n.n)

［功能］　返回浮点表达式 n.n(n.n>0)的自然对数值。

［示例］　SELECT　LOG(4.78)=1.56444054650336

SELECT　LOG(0.25)=−1.38629436111989

15）LOG10

［格式］　LOG10(n.n)

［功能］　返回以 10 为底的浮点表达式 n.n 的对数值。

［示例］　SELECT　LOG10(4.78)=0.679427896612119

SELECT　LOG10(0.25)=−0.602059991327962

16）PI

［格式］　PI()

［功能］　返回圆周率π的值。

［示例］　SELECT　PI()=3.14159265358979

17）POWER()

［格式］　POWER(m,n)

［功能］　返回 m 的 n 次方的值。

［示例］　SELECT　POWER(2,4)=16

SELECT　POWER(6.5,2)=42.25

18）RADIANS

［格式］　RADIANS(n)

［功能］　将指定的度数 n 转换为弧度。

［示例］　SELECT　RADIANS(42.97)=0.749967979581963420

SELECT　RADIANS(90)=1

19）RAND

［格式］　RAND([n])

［功能］　返回一个随机的 0 到 1 之间的 FLOAT 类型的数值。

20）ROUND

［格式］　ROUND(n,p[t])

［功能］　对数值表达式 n 进行四舍五入，p 表示舍入的精度。如果 p 是正数，就对小数点右边的数字进行四舍五入；如果 p 是负数，就对小数点左边的数字进行四舍五入。如果省略可选参数 t 或其值为 0（默认值），则将传入 n，如果指定了 0 以外的值，则将截断 n。

［示例］　SELECT　ROUND(3.6567,3)=3.6570

SELECT　ROUND(748.58,−2)=700.00

SELECT　ROUND(345.4567,−1,1)=340.0000

21）SIGN

［格式］　SIGN(n)

［功能］　返回数字表达式 n 的符号数字。正数返回 1，负数返回−1，0 返回 0。

［示例］　SELECT　SIGN(3.14)=1

SELECT　SIGN(−3.14)=−1

SELECT　SIGN(0)=0

22）SQUARE

［格式］　SQUARE(n)

［功能］　返回数字表达式 n 的平方值。

［示例］　SELECT　SQUARE(5)=25

SELECT　SQUARE(4.5)=20.25

23）SQRT

［格式］　SQRT(n)

［功能］　返回数字表达式 n 的平方根。

［示例］　SELECT　SQRT(25)=5

SELECT　SQRT(20.25)=4.5

2．字符串函数

字符串函数主要用于对字符串进行连接、截取等操作。

1）ASCII

［格式］　ASCII（字符表达式）

［功能］　返回字符表达式最左边字符的 ASCII 码。

［示例］　SELECT　ASCII('A')=65

SELECT　ASCII('abcd')=97

2）CHAR

［格式］　CHAR（整型表达式）

［功能］　将整型表达式所对应的 ASCII 值转换为字符。

［示例］　SELECT　CHAR(65)='A'

SELECT　CHAR(97)='a'

3）CHARINDEX

［格式］　CHARINDEX(x,y)

［功能］　返回字符串 x 在字符串 y 中首次出现的起始位置，如果没有出现，则返回值 0。

［示例］　SELECT　CHARINDEX('basic', 'visual basic')=8

4）DIFFERENCE

［格式］　DIFFERENCE(x,y)

［功能］　返回的整数是 SOUNDEX 值中（用于评估两个字符串的相似性，具体可参阅本节字符串函数中第（17）个函数）相同字符的个数。返回的值从 0～4 不等，0 表示几乎不同或完全不同，4 表示几乎相同或完全相同。

［示例］　SELECT　DIFFERENCE('bit', 'bet')=3

5）LEFT

［格式］　LEFT(x,n)

［功能］　返回字符串 x 中最左边的 n 个字符。

［示例］　SELECT　LEFT('basic',3)='bas'

6）LEN

［格式］　LEN(x)

［功能］　返回字符串 x 的字符个数，而不是字节个数，包括空格的个数。

［示例］　SELECT　LEN('ABCDEFG') = 7

7）LOWER

［格式］　LOWER(x)

［功能］　将字符串表达式 x 中的大写字母转换成小写字母返回，其中的小写字母和非字母字符保持不变。

［示例］　SELECT　LOWER('aAbB123') = 'aabb123'

8）LTRIM

［格式］　LTRIM(x)

［功能］　将字符串表达式 x 左边的前导空格去掉后返回。

［示例］　SELECT　LTRIM('　abcd') = 'abcd'

9）NCHAR

［格式］　NCHAR(n)

［功能］　返回整型表达式 n 所代表的 Unicode 字符。

［示例］　SELECT　NCHAR(65) = 'A'

10）QUOTENAME

［格式］　QUOTENAME(x[,t])

［功能］　返回添加了分隔符的 Unicode 字符串。其中 x 表示待操作的字符串，t 表示分隔符，可以是单引号（'）、左方括号或右方括号（[]），以及英文双引号（""）。如果没有指定 t，则使用双引号。

［示例］　SELECT　QUOTENAME('ABC DEF') = [ABC DEF]

11）PATINDEX

［格式］　PATINDEX(%p%,x)

［功能］　返回指定模式 p 在字符串 x 中第一次出现的位置，如果没有出现，则返回值 0。

［示例］　SELECT　PATINDEX('%cd%', 'abcdefg') = 3

12）REPLACE

［格式］　REPLACE(x,m,n)

［功能］　将字符串 x 中出现 m 的地方替换成 n 后返回。

［示例］　SELECT　REPLACE('bad', 'b', 's') = 'sad'

13）REPLICATE

［格式］　REPLICATE(x,n)

［功能］　以指定的次数 n 重复字符串 x 后返回。

［示例］　SELECT　REPLICATE('ABC',3) = 'ABCABCABC'

14）REVERSE

［格式］　REVERSE(x)

［功能］　将字符串 x 显示为逆序。

［示例］　SELECT　REVERSE('ABCD') = 'DCBA'

15）RIGHT

［格式］　RIGHT(x,n)

［功能］　返回字符串 x 最右边的 n 个字符。

［示例］　SELECT　RIGHT('ABCD',2) = 'CD'

16）RTRIM

［格式］　RTRIM(x)

［功能］　将字符串 x 的尾部空格去掉后返回。

［示例］　SELECT　RTRIM('ABCD　　') = 'ABCD'

17）SOUNDEX

［格式］　SOUNDEX(x)

［功能］　返回字符串 x 所对应的 4 个字符组成的代码，用于评估两个字符串的相似性。

［示例］　SELECT　SOUNDEX('ABCD') = A120

18）SPACE

［格式］　SPACE(n)

［功能］　返回 n 个空格表示的字符串。

19）STR

［格式］　STR(n[len[,t]])

［功能］　将指定的浮点表达式 n 转换成字符串返回。len 表示返回字符串的长度，包括数字、正负号、小数点和空格，默认值为 10，t 表示小数点右边的被返回的数字。

［示例］　SELECT　STR(1.23456,4,2) = '1.23'

20）STUFF

［格式］　STUFF(x,start,len,y)

［功能］　将字符串 x 中 start 位置开始的 len 个字符替换成字符串 y 返回。

［示例］　SELECT　STUFF('ABCD',2,3, 'X') = 'AXXX'

21）SUBSTRING

［格式］　SUBSTRING(x,start,len)

［功能］　返回字符串 x 中从 start 位置开始的 len 个字符。

［示例］　SELECT　SUBSTRING('helloworld',1,5) = 'hello'

22）UNICODE

［格式］　UNICODE(x)

［功能］　返回字符串 x 最左边字符的 Unicode 代码。

［示例］　SELECT　UNICODE('A') = 65

23）UPPER

［格式］　UPPER(x)

［功能］　将字符串表达式 x 中的小写字母转换成大写字母返回，其中的小写字母和非字母字符保持不变。

［示例］　SELECT　LOWER('aAbB123') = 'AABB123'

3．日期函数

日期函数可用来显示日期和时间的信息。

1）GETDATE

［格式］　GETDATE()

［功能］　返回系统当前的日期和时间。

［示例］　SELECT　GETDATE() = 2009-10-11 17:52:31.027

2）DATEPART

［格式］　DATEPART(item,date)

［功能］　返回日期指定的 item，其中 date 为一个整数。

［示例］　SELECT　DATEPART(weekday, '01.01.2010') = 6

SELECT　DATEPART(month, '01.01.2010') = 1

3）DATENAME

［格式］　DATENAME(item,date)

［功能］　返回指定日期的名字，返回值是一个字符串。

［示例］　SELECT　DATENAME(weekday, '01.01.2010') = '星期五'

4）DATEDIFF

［格式］　DATEDIFF(item,startdate,enddate)

［功能］　返回指定的 startdate 和 enddate 之间所跨的指定 item 边界的计数（带符号的整数）。

［示例］　SELECT　DATEDIFF(year, '2005-12-31 23:59:59.9999999' , '2006-01-01 00:00:00.0000000') = 1

5）DATEADD

［格式］　DATEADD(datepart,number,date)

［功能］　将指定的 number 时间间隔（有符号整数）与指定 date 的指定 datepart 相加后，返回该 date。

［示例］　SELECT　DATEADD(month,1, '2010-01-01') = '2010-02-01 00:00:00.000'

4．系统函数

系统函数提供了一些有关数据库系统的信息。

1）APP_NAME

［格式］　APP_NAME()

［功能］　返回当前会话的应用程序名称（如果应用程序进行了设置）。

2）CAST

［格式］　CAST(expression　AS　data_type[(length)])

［功能］　返回转换为 data_type 的 expression。

3）COALESCE

［格式］　COALESCE(expression[,…n])

［功能］　返回其参数中第一个非空表达式。

4）COL_LENGTH

［格式］　COL_LENGTH(table_name,column_name)

［功能］　返回 table_name 表中 column_name 列的定义长度，以字节为单位。

5）CONVERT

［格式］　CONVERT(data_type[(length)],expression[,style])

［功能］　返回转换为 data_type 的 expression。其中，可选参数 length 指定目标数据类型长度的可选整数，默认值为 30。可选参数 style 指定 CONVERT 函数如何转换 expression 的整数表达式。如果样式为 NULL，则返回 NULL。该范围是由 data_type 确定的。

6）CURRENT_TIMESTAMP

［格式］　CURRENT_TIMESTAMP

［功能］　返回当前数据库系统时间戳，返回值的类型为 DATATIME，并且不含数据库时区偏移量。

［示例］　SELECT　CURRENT_TIMESTAMP = '2010-01-01　12:12:12.670'

7）CURRENT_USER

［格式］　CURRENT_USER

［功能］　返回当前用户的名称，等价于函数 USER_NAME()。

8）DATALENGTH

［格式］　DATALENGTH(expression)

［功能］　返回表达式所占用的字节数。

9）GETANSINULL

［格式］　GETANSINULL(['database_name'])

［功能］　如果按照 ANSI　SQL 标准在数据库 database_name 中使用 NULL 值，那么返回值为 1。

［示例］　SELECT　GETANSINULL('AdventureWorks') = 1

10）ISNULL

［格式］　ISNULL(check_expression,replacement_value)

［功能］　检查 check_expression 是否为 NULL，如果为 NULL，则返回值 replacement_value，如果不为 NULL，则返回值 check_expression。

11）ISNUMERIC

［格式］　ISNUMERIC(expression)

［功能］　判断表达式是否为有效的数值类型。

12）NEWID

［格式］　NEWID()

［功能］　创建 uniqueidentifier 类型的全局唯一标志符。

13）NEWSEQUENTIALID

［格式］　NEWSEQUENTIALID()

［功能］　在指定计算机上指定 GUID（全局唯一标志符），该 GUID 值比之前通过该函数产生的任何 GUID 值都大。如果涉及保密问题，则不要使用该函数。可以试着猜想下一个生成的 GUID 的值，从而访问与该 GUID 关联的数据。

14）NULLIF

［格式］　NULLIF(expression1,expression2)

［功能］　如果表达式 expression1 和表达式 expression2 相等，则返回 NULL 值。

15）SERVERPROPERTY

［格式］　SERVERPROPERTY(propertyname)

［功能］　返回有关服务器实例的属性信息。

16）SYSTEM_USER

［格式］　SYSTEM_USER

［功能］　返回目前用户的登录 ID。

17）USER_ID

［格式］　USER_ID(['user_name'])

［功能］　返回数据库用户 user_name 的标志号。如果没有指定名字，则检索当前用户的标志符。

［示例］　SELECT　USER_ID('guest') = 2

18）USER_NAME

［格式］　USER_NAME([id])

［功能］　返回基于指定的标志号的数据库用户名。如果没有指定用户名字，则检索当前用户的名字。

［示例］　SELECT　USER_NAME(2) = 'guest'

5．元数据函数

一般而言，元数据函数返回的是有关指定数据库和数据对象的信息。

1）COL_NAME

［格式］　COL_NAME(table_id,column_id)

［功能］　根据指定的对应表标志号和列标志号返回列的名称。

2）COLUMNPROPER

［格式］　COLUMNPROPER(id,column,property)

［功能］　返回有关列或过程参数的信息。

3）DATABASESEPROPERTY

［格式］　DATABASESEPROPERTY(database,property)

［功能］　返回指定数据库和属性名的命名数据库属性值。

4）DB_ID

［格式］　DB_ID(['database_name'])

［功能］　返回数据库的标志（ID）号。

5）DB_NAME

［格式］　DB_NAME(database_id)

［功能］　返回数据库名称。

6）INDEX_COL

［格式］　INDEX_COL(table,i,no)

［功能］　返回表 table 中的索引列，该索引列由索引标志符 i 及该列在索引中的位置 no 指定。

7）INDEXPROPERTY

［格式］　INDEXPROPERTY(object_id,index_or_statistics_name,property)

［功能］　根据指定的表标志号、索引或统计信息名称及属性名称，返回已命名的索引或统计信息属性值。对于 XML 索引，返回 NULL。

8）OBJECT_NAME

［格式］　OBJECT_NAME(object_id[,database_id])

［功能］　返回架构范围内对象的数据库对象名称。

9）OBJECT_ID

［格式］　OBJECT_ID(object_name)

［功能］　返回数据库对象 object_name 的标志符。

10）OBJECTPROPERTY

［格式］　OBJECTPROPERTY(id,property)

［功能］　返回当前数据库中架构范围内的对象的有关信息。

5.5　用户定义函数

为了扩展 Transact-SQL 语言的编程能力，SQL Server 2008 除了提供很多系统内置的函数以外，还提供了用户定义函数。与其他编程语言中的函数类似，Microsoft SQL Server 用户定义函数也是接受参数、执行操作（如复杂计算）并将操作结果以值的形式返回的例程。返回值可以是单个标量值或结果集。

5.5.1　用户定义函数的定义与调用

1．用户定义函数的定义

用户定义函数（User-defined Functions UDFS）可使用 CREATE　FUNCTION 的格式创建，语法格式如下：

```
CREATE　FUNCTION[schema_name.]function_name
    [({@param }　type　[= default]) {…}
    RETURNS　{ scalar_type | [@variable] TABLE }
    [WITH　{ ENCRYPTION | SCHEMABINDING }
    [AS]
    BEGIN
  function_body
      RETURN　scalar_expression
    END
```

参数说明如下。

- schema_name：用户定义函数的创建者所指定的表的名字。
- function_name：用户定义函数名。该函数的名称必须符合标志符的规则，它在数据库中必须是唯一的。
- @param：用户定义函数的形参名。CREATE FUNCTION 语句中可以同时声明一个或多个形参名，用符号@作为第一个字符来指定参数的名称，每个函数的参数的作用范围仅限于该函数。
- default：指定对应参数的默认值（默认值也可以为 NULL 值）。
- RETURES 从句：指定了用户定义函数返回值的数据类型。该数据类型几乎可以是数据库系统所支持的所有标准的数据类型（不包括数据类型 TIMESTAMP）。
- WITH ENCRYPTION：对函数体进行加密。使用 ENCRYPTION 可以避免将函数作为 SQL Server 复制的一部分发布。
- WITH SCHEMABINDING：用于将创建的函数与数据库对象绑定。绑定以后，则不能修改或删除该函数引用的数据库对象。只有当该函数被删除的时候，绑定才会被解除。
- function_body：由一系列 Transact-SQL 语句序列组成的函数体。
- scalar_expression：指定标量函数返回的数量值。

【例 5.9】　创建一个自定义函数，返回全体学生某门功课的平均成绩。

```
CREATE  FUNCTION  average_grade(@cno  char(18))  RETURNS  int
AS
BEGIN
   DECLARE  @avggrade  int
   SELECT  @avggrade =
      ( SELECT  avg(Grade)
        FROM  SC
        WHERE  Cno = @cno
        GROUP  BY  Cno
      )
 RETURN  @avggrade
END
```

说明：在 Transact-SQL 中声明变量都是以 DECLARE 关键字开头的。

在【对象资源管理器】面板中选择服务器，展开数据库选项【db_stu】，单击【可编程性】选项并展开，单击【函数】选项并展开，最后再单击【标量值函数】选项并展开，这时我们可以看到刚才创建的自定义函数【average_grade】，如图 5-5 所示。

2．用户定义函数的调用

任何一个用户定义的函数，都可以用 Transact-SQL 语句调用，如 SELECT 语句、INSERT 语句、UPDATE 语句和 DELETE 语句。调用函数的时候，先写出函数的名字，然后跟上一对圆括号。在圆括号里，可以指定一个或多个参数，参数可以是值，也可以是表达式的形式。调用函数的时候，如果没有对参数指定默认值，就必须按照定义的时候参数声明的顺序依次给其赋值。

图 5-5　查看标量值函数【average_grade】

【例 5.10】　调用例 5.9 中创建的函数。

```
SELECT　dbo.average_grade('3')
```

结果如图 5-6 所示。

	(无列名)
1	82

图 5-6　调用函数 average_grade

5.5.2　用户定义函数的删除

对于一个已经创建的用户定义函数，可以有两种方法删除。

1．使用对象资源管理器删除用户定义函数

在【Sql Server Management Studio】的【对象资源管理器】面板中选择服务器，展开数据库选项【db_stu】，单击【可编程性】选项并展开，单击【函数】选项并展开，最后再单击【标量值函数】选项并展开，右击要删除的用户定义函数的名称，在弹出的快捷菜单中选择【删除】命令，如图 5-7 所示。

图 5-7　删除函数【average_grade】

2．使用 Transcat-SQL 语句删除用户定义函数

利用 Transact-SQL 语句 DROP FUNCTION 可删除用户定义函数，语法格式如下：

```
DROP　FUNCTION{[owner_name.]function_name}[,…n]
```

参数说明如下。

- owner_name：所有者名。
- function_name：指要删除的用户定义函数的名称。
- n：表示可以指定多个用户定义函数，并予以删除。

习　　题

1．创建一个自定义函数，返回某个专业学生某门课程的平均分。

2．试说明系统内置函数的分类及各类函数的特点。

3．在 SQL Server 2008 中，下列变量名正确的是（　　）。

A．@sum　　B．j

C．sum　　D．4kk

4．下列标志符可以作为局部变量使用的是（　　）。

A．Myvar　　B．My var

C．@Myvar　　D．@My var

5．T-SQL 语言中行注释的符号为_____，块注释的符号为_____。

6. SQL Server 中的变量分为两种，全局变量和局部变量。其中全局变量的名称以_____字符开始，由系统定义和维护。局部变量以_____字符开始，由用户自己定义和赋值。

第 6 章　索引与数据完整性

- 了解索引的分类和特点
- 掌握索引的创建、查看和删除方法
- 了解数据完整性的概念和实现方法

6.1　索　　引

索引是与表或视图关联的磁盘上的结构，它可以加快从表或视图中检索行的速度。索引包含由表或视图中的一列或多列生成的键。这些键存储在一个结构（B 树）中，使 SQL Server 可以快速有效地查找与键值关联的行。

6.1.1　索引的分类

表或视图可以包含以下类型的索引。

1．聚集索引

聚集索引根据数据行的键值在表或视图中排序和存储这些数据行。索引定义中包含聚集索引列。每个表只能有一个聚集索引，因为数据行本身只能按一个顺序排序。只有当表包含聚集索引时，表中的数据行才按排序顺序存储。如果表具有聚集索引，则该表称为聚集表。如果表没有聚集索引，则其数据行存储在一个称为堆的无序结构中。

2．非聚集索引

非聚集索引具有独立于数据行的结构。非聚集索引包含非聚集索引键值，并且每个键值项都有指向包含该键值的数据行的指针。从非聚集索引中的索引行指向数据行的指针称为行定位器。行定位器的结构取决于数据页是存储在堆中还是聚集表中。对于堆，行定位器是指向行的指针。对于聚集表，行定位器是聚集索引键。

6.1.2　索引的创建

当执行查询时，如果有索引，查询优化程序将首先使用索引，但索引会减慢数据修改的速度，因此对于是否建立索引，用户需要根据实际情况判断。

通常，适合创建索引的列如下：

- 主键及外键所在的列；
- 频繁作为 WHERE 子句条件使用的列；
- 经常在 ORDER BY 子句中使用的列；
- 取值唯一的列。

不适合创建索引的列如下：

- 查询中 WHERE 子句不用或很少用到的列；
- 列的取值只有 1～2 个或包含太多重复值；
- 表中的数据少；
- 维护索引的价值不大。

在 SQL Server 中可以通过对象资源管理器和 T-SQL 语句创建索引。下面将分别介绍。

1．使用对象资源管理器创建索引

下面以表 Student 为例来介绍创建索引的方法，具体步骤如下。

（1）在【对象资源管理器】中找到要创建索引的数据库【db_stu】中的表【Student】，右击，在弹出菜单中选择【设计】命令，打开表【Student】，选中要设置索引的列，右击，在弹出菜单中选择【索引/键】命令或在菜单栏的【表设计器】中选择【索引/键】命令，如图 6-1 所示。

（2）在弹出的【索引/键】对话框中，单击 添加(A) 按钮，为该列添加索引，在右边的【常规】项中可以设置该索引的属性，如图 6-2 所示。

（3）设置完成后，单击 关闭(C) 按钮，完成索引的创建。

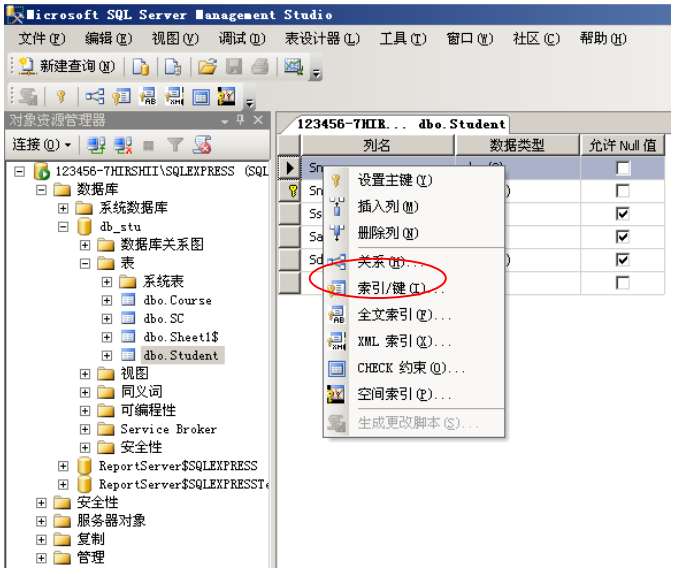


图 6-1　选择【索引/键】命令

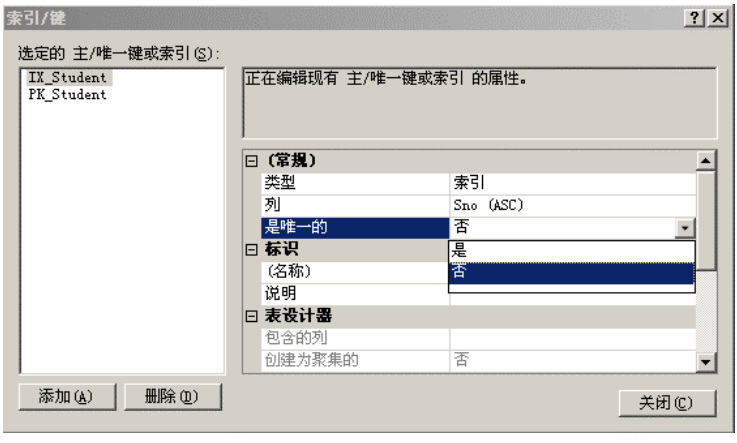


图 6-2　创建索引

2．使用 T-SQL 语句创建索引

在 T-SQL 语句中，可以使用 CREATE INDEX 语句来创建索引，语法格式如下：

```
CREATE [UNIQUE][CLUSTERED|NONCLUSTERED] INDEX index_name ON table_name(column_name [ASC|DESC][,…n])
```

其中，各参数的含义如下。

- UNIQUE：为表或视图建立唯一索引。
- CLUSTERED|NONCLUSTERED：用于建立聚集或非聚集索引。
- index_name：索引名称。
- table_name：指定建立索引的表名。
- column_name：指定建立索引的列名。

【例 6.1】　下面的语句用于在表【Student】的【Sno】列上建立非聚集索引，索引名字为【stu_index】，顺序为升序。

```
USE db_stu
GO
CREATE NONCLUSTERED
  INDEX stu_index
On Student(Sno ASC)
```

6.1.3　索引的删除

索引会影响修改或删除记录的速度，因此，当不需要索引时，就应当及时删除。索引的删除可以使用对象资源管理器，也可以使用 T-SQL 语句。

1．使用对象资源管理器

使用对象资源管理器删除索引的具体操作步骤如下。

（1）在【对象资源管理器】中找到要删除索引的表，在该表的折叠项中找到【索引】，展开【索引】，选中要删除的索引，右击，在弹出的菜单中选择【删除】命令。

（2）在弹出的【删除对象】对话框中，单击 确定 按钮，即可删除索引。

2．使用 T-SQL 语句

在 T-SQL 语句中，可以使用 DROP INDEX 语句删除数据库中相应表的索引，其语法格式如下：

```
DROP INDEX table_name.index_name[,…n]
```

【例 6.2】　下面的语句用于从表【Student】中删除索引【stu_index】。

```
DROP INDEX Student.stu_index
```

6.2　默认值约束及默认值对象

对于某些字段，可在程序中定义默认值。一个字段默认值的建立可通过如下两种形式来实现：

- 在定义表或修改表时，定义默认值约束；
- 先定义默认对象，然后将该对象绑定到表的相应字段。

6.2.1　在表中定义及删除默认值约束

在 SQL Server 中，DEFAULT 约束用于定义列的默认值，可以通过 SQL Server 管理控制台和 T-SQL 语句来实现。

1．使用 SQL Server 管理控制台

具体操作步骤如下。

（1）在【对象资源管理器】中，右击要更改其小数位数的列所在的表，再单击【设计】命令。此时，将在表设计器中打开该表。

（2）选择要为其指定默认值的列。

（3）在【列属性】选项卡的【默认值或绑定】属性中输入新的默认值，如图 6-3 所示。

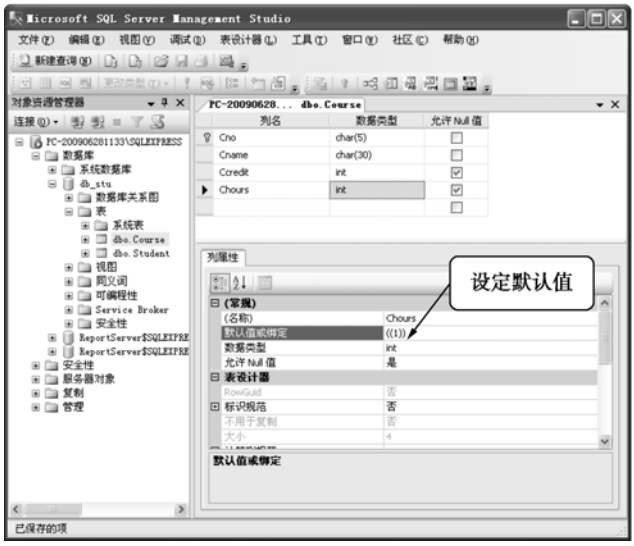


图 6-3　定义 DEFAULT 约束

2．使用 T-SQL 语句

在 T-SQL 语句中，可以使用 ALTER TABLE 语句定义 DEFAULT 约束，其语法格式如下：

```
ALTER TABLE table_name
ADD CONSTRAINT default_name DEFAULT values
```

其中，table_name 为表名称，ADD CONSTRAINT 表示在已定义的表 table_name 中增加一个约束定义，default_name 为指定的约束名称。

【例 6.3】　下面的语句用于定义表【Ccourse】中【Chours】列的 DEFAULT 约束。

```
USE db_stu
ALTER TABLE Ccourse
ADD CONSTRAINT Chours DEFAULT 1
```

DEFAULT 约束的删除可以使用 DROP DEFAULT 语句实现。

【例 6.4】　下面的语句用于删除【Chours】列约束。

```
USE db_stu
ALTER TABLE Ccourse
DROP DEFAULT Chours
```

6.2.2　默认值对象的定义、使用与删除

默认值对象的定义和使用可用 T-SQL 语句来实现。

1．通过 T-SQL 语句定义和绑定 DEFAULT 默认值对象

通过 T-SQL 语句定义 DEFAULT 默认值对象的语法格式如下：

```
CREATE DEFAULT [ schema_name .] default_name AS constant_expression [ ; ]
```

其中，各参数的含义如下。

- schema_name：默认值所属架构的名称。
- default_name：默认值的名称。默认值名称必须遵守标志符规则。可以选择是否指定默认值所有者名称。
- constant_expression：只包含常量值的表达式，除了那些包含别名数据类型的表达式外，可以使用任何常量、内置函数或数学表达式。不能使用用户定义函数。字符和日期常量要放在单引号（'）内；货币、整数和浮点常量不需要引号。二进制数据必须以 0x 开头，货币数据必须以美元符号（$）开头。默认值必须与列数据类型兼容。

通过系统存储过程绑定 DEFAULT 默认值对象，可以使用 sp_bindefault 语句来实现，语法格式如下：

```
sp_bindefault [ @defname = ] 'default' ,
[ @objname = ] 'object_name' [ , [ @futureonly = ] 'futureonly_flag' ]
```

其中各参数的含义如下。

- [@defname =] 'default'：由 CREATE DEFAULT 创建的默认值的名称。default 的数据类型为 nvarchar(776)，没有默认值。
- [@objname =] 'object_name'：表名和列名或者绑定默认值的别名数据类型。object_name 的数据类型为 nvarchar(776)，没有默认值。object_name 不能使用 varchar(max)、nvarchar(max)、varbinary(max)、xml 或 CLR 用户定义类型进行定义。
- [@futureonly =] 'futureonly_flag'：只有将默认值绑定到别名数据类型时才使用。futureonly_flag 的数据类型为 varchar(15)，默认值为 NULL。当此参数设置为 futureonly 时，该数据类型的现有列无法继承新默认值。将默认值绑定到列时，从不使用此参数。如果 futureonly_flag 为 NULL，则新默认值将绑定到别名数据类型的所有列，这些列当前没有默认值或正在使用别名数据类型的现有默认值。

【例 6.5】　下面的语句可以将数据库【db_stu】中表【Course】中字段【Ccredit】的初始值设定为 0。

```
CREATE DEFAULT xf_default AS 0
USE db_stu
EXEC　sp_bindefault 'xf_default','Course.Ccredit'
```

```
GO
```

2．默认值对象的删除

可以在对象资源管理器中删除默认值对象，也可以用 T-SQL 语句删除。注意，要删除默认值对象首先要解除它与表字段的绑定关系。

1）利用 sp_unbindefault 解除绑定关系

其语法格式如下：

```
sp_unbindefault [ @objname = ] 'object_name' [ , [ @futureonly = ] 'futureonly_flag' ]
```

其各参数的含义如下。

- [@objname =] 'object_name'：解除其默认值绑定的表和列或别名数据类型的名称。object_name 的数据类型为 nvarchar(776)，无默认值。SQL Server 尝试先将两部分的标志符解析为列名，再解析为别名数据类型。
- [@futureonly =] 'futureonly_flag'：仅在解除别名数据类型的默认值绑定时使用。futureonly_flag 的数据类型为 varchar(15)，默认值为 NULL。当 futureonly_flag 的数据类型为 futureonly 时，该数据类型的现有列不会失去指定默认值。

2）删除默认值对象

可以用 DROP 语句删除，语法格式如下：

```
DROP DEFAULT {default}[,…n]
```

【例 6.6】 下面语句可以解除默认值对象 xf_default 与【db_stu】数据库中表【Ccourse】中学分字段的绑定，并将此对象删除。

```
USE db_stu
EXEC    sp_inbindefault 'Course.Ccredit'
DROP DEFAULT xf_default
```

6.3　数据完整性

数据完整性是指存储在数据库中的数据的一致性和正确性。

6.3.1　数据完整性的分类

根据作用域的数据库对象和范围的不同，数据完整性可以分为如下几类。

1．实体完整性

实体完整性将行定义为特定表的唯一实体。实体完整性通过 UNIQUE 索引、UNIQUE 约束或 PRIMARY KEY 约束，强制表的标志符列或主键的完整性。

2．域完整性

域完整性指特定列的项的有效性。用户可以强制域完整性限制类型（通过使用数据类型）、限制格式（通过使用 CHECK 约束和规则）或限制可能值的范围（通过使用 FOREIGN KEY 约束、CHECK 约束、DEFAULT 定义、NOT NULL 定义和规则）。

3．引用完整性

输入或删除行时，引用完整性保留表之间定义的关系。在 SQL Server 中，引用完整性通

过 FOREIGN KEY 和 CHECK 约束，以外键与主键之间或外键与唯一键之间的关系为基础。引用完整性确保键值在所有表中一致。这类一致性要求不引用不存在的值，如果一个键值发生更改，则整个数据库中，对该键值的所有引用都要进行一致的更改。

4．用户定义完整性

用户定义完整性使用户可以定义不属于其他任何完整性类型的特定业务规则。所有完整性类型都支持用户定义完整性，这包括 CREATE TABLE 中所有列级约束和表级约束、存储过程，以及触发器。

6.3.2　域完整性的实现

通过定义约束等对象可以实现域完整性。

CHECK 约束用于规范列的取值，列值必须满足约束的范围，范围外的数据无法输入。CHECK 约束的定义和删除可以通过 SQL Server 管理控制台和 T-SQL 语句来实现。下面通过这两种方法将学生课程表学分（Ccredit）列的取值范围定义为 0～20。

1．使用 SQL Server 管理控制台

具体操作步骤如下。

（1）在对象资源管理器中打开表【Ccourse】的表设计器窗口，在列名中选择【Ccredit】列，单击鼠标右键，在弹出的快捷菜单中选择【CHECK 约束】命令，如图 6-4 所示。

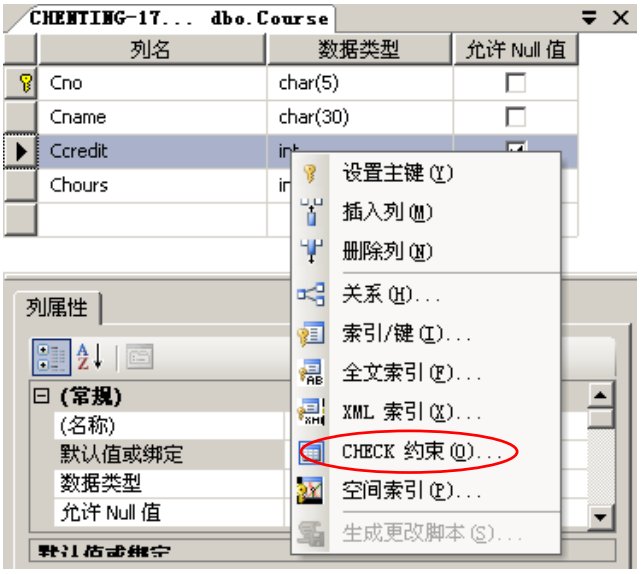


图 6-4　选择【CHCKE 约束】命令

（2）在打开的【CHECK 约束】对话框中单击 添加(A) 按钮，在【名称】中输入要定义约束的列的名字，在【表达式】中直接输入条件表达式。如图 6-5 所示，单击 ... 按钮，在弹出的【CHECK 约束表达式】对话框中输入条件表达式，然后单击 确定 按钮，以完成表达式的输入，如图 6-6 所示。单击 关闭(C) 按钮，完成【Ccredit】列的 CHECK 约束定义。

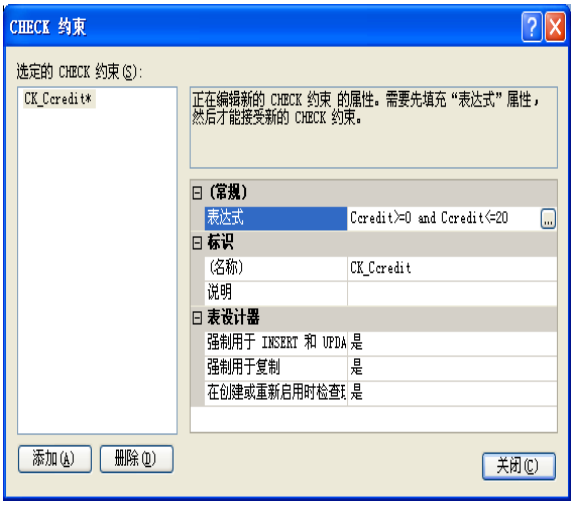


图 6-5　输入条件表达式和列名称

图 6-6　【CHECK 约束表达式】对话框

（3）如果要删除约束，只需在【CHECK 约束】对话框中选中要删除的约束，单击 删除(D) 按钮即可删除约束，如图 6-7 所示。

图 6-7　删除【CHECK 约束】

2．使用 T-SQL 语句

在 T-SQL 语句中，可以使用 ALTER TABLE 语句定义 CHECK 约束，其语法格式如下：

```
ALTER TABLE table_name
ADD CONSTRAINT check_name CHECK (logical_expression)
```

其中，table_name 为表名称，ADD CONSTRAINT 表示在已定义的表 table_name 中增加一个约束定义，check_name 为指定的约束名称，logical_expression 为 CHECK 约束表达式。

【例 6.7】　下面的语句用于定义表【Ccourse】中【Ccredit】列的 CHECK 约束。

```
USE db_stu
ALTER TABLE Ccourse
ADD CONSTRAINT Ccredit CHECK(Ccredit>=0 and Ccredit<=20)
```

CHECK 约束的删除可以使用 DROP CONSTRAINT 语句实现。例如，下面的语句用于删除【Ccredit】列约束。

```
USE db_stu
ALTER TABLE Ccourse
DROP CONSTRAINT Ccredit
```

6.3.3 实体完整性的实现

实体完整性可以通过建立唯一索引、UNIQUE 约束、PRIMARY KEY 约束及 IDENTTTY 属性来实现。

1．PRIMARY KEY 约束

表一般有一列或一组列用于唯一标识表中每一行的值。这样的一列或多列称为表的主键（PK），用来强制表的实体完整性。在创建或修改表时，可以通过定义 PRIMARY KEY 约束来创建主键。一个表只能有一个 PRIMARY KEY 约束，并且 PRIMARY KEY 约束中的列不能接受空值。由于 PRIMARY KEY 约束可保证数据的唯一性，因此应经常对标志列定义这种约束。

如果为表定义了 PRIMARY KEY 约束，则数据库引擎将通过为主键列创建唯一索引来强制数据的唯一性。当在查询中使用主键时，此索引还可用来对数据进行快速访问。因此，所选的主键必须遵守创建唯一索引的规则。在 SQL Server 2008 中可以使用 SQL Server 管理控制台定义和修改 PRIMARY KEY 约束。

注意：使用 SQL Server 管理控制台定义 PRIMARY KEY 约束在 2.2.1 节定义主关键字中已叙述过，这里不再叙述。下面只介绍使用 T-SQL 语言定义和修改 PRIMARY KEY 约束。

在 T-SQL 语句中，可以使用 CREATE TABLE 语句创建 PRIMARY KEY 约束，语法格式如下：

```
CREATE TABLE table_name
(column_name datetype [CONSTRAINT constraint_name] NOT NULL PRIMARY [CLUSTERED|NONCLUSTERED][,...n])
```

其中，各参数的含义如下。

- table_name：指定的表名。
- column_name：指定的列名。
- CONSTRAINT：指定约束名 constraint_name。
- PRIMARY KEY：定义约束类型。
- CLUSTERED|NONCLUSTERED：定义约束的索引类型。

【例 6.8】　下面的语句用于创建表【Course】，并把【Cno】列定义为 PRIMARY KEY 约束。

```
USE db_stu
CREATE TABLE Course
(Cno char(5)PRIMARY KEY NOT NULL,
Cname char(30)NOT NULL,
```

```
Ccredit int NULL,
Chours int NULL
)
```

也可以使用 ALTER TABLE 语句为已存在的表创建 PRIMARY KEY 约束，其语法格式如下：

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name]
PRIMARY KEY CLUSTERED|NOTCLUSTERED(column[,...n])
```

【例 6.9】　下面的语句用于把表【Course】中的【Cno】列定义为 PRIMARY KEY 约束。

```
USE db_stu
ALTER TABLE Course
ADD CONSTRAINT Cno
PRIMARY KEY CONSTRAINT
GO
```

可以使用 DROP 语句删除 PRIMARY KEY 约束，其语法格式如下：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name[,...n]
```

【例 6.10】　下面的语句用于删除已有的 PRIMARY KEY 约束。

```
USE db_stu
ALTER TABLE Course
DROP CONSTRAINT Cno
GO
```

2．UNIQUE 约束

UNIQUE 约束可以确保在非主键列中不输入重复的值。尽管 UNIQUE 约束和 PRIMARY KEY 约束都强制唯一性，但想要强制一列或多列组合（不是主键）的唯一性时应使用 UNIQUE 约束而不是 PRIMARY KEY 约束。一个表中可以定义多个 UNIQUE 约束，但却只能定义一个 PRIMARY KEY 约束。而且，UNIQUE 约束允许 NULL 值，这一点与 PRIMARY KEY 约束不同。不过，当与参与 UNIQUE 约束的任何值一起使用时，每列只允许一个空值。FOREIGN KEY 约束可以引用 UNIQUE 约束。

UNIQUE 约束可以用以下两种方法定义。

1）使用 SQL Server 管理控制台定义和修改 UNIQUE 约束

使用 SQL Server 管理控制台定义和修改 UNIQUE 约束的具体步骤如下。

（1）在【对象资源管理器】中，右击要为其添加唯一约束的【Course】表，再单击【设计】命令，将在表设计器中打开该表。

（2）在菜单栏上单击【表设计器】选项，在下拉菜单中单击【索引/键】命令，如图 6-8 所示。或在将要定义或修改 UNIQUE 约束的列名上单击鼠标右键，在下拉菜单中单击【索引/键】命令，如图 6-9 所示。系统弹出【索引/键】对话框，如图 6-10 所示。

（3）在【索引/键】对话框中，单击【添加】按钮。在【常规】中单击【类型】选项，再从右侧的属性下拉列表框中选择【唯一键】选项，然后再单击【列】选项，如图 6-11 所示。

（4）单击属性右侧的按钮，从弹出的【索引列】对话框中选择将要添加约束的列名，如图 6-12 所示。当保存表时，即会在数据库中创建该唯一约束。如果要改变该列的约束，可以在第（3）步中从下拉列表框中选择其他约束。

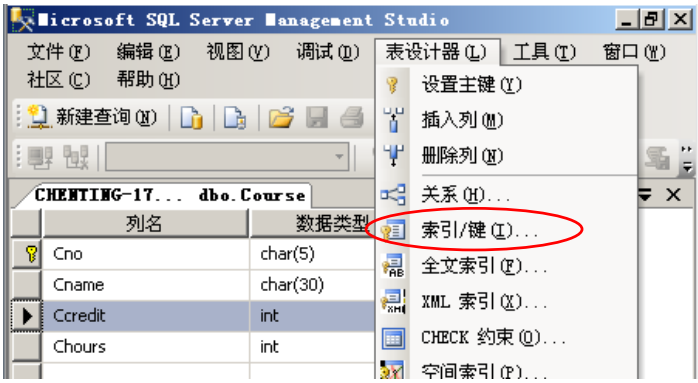


图 6-8　选择【索引/键】命令

图 6-9　右击选择【索引/键】命令

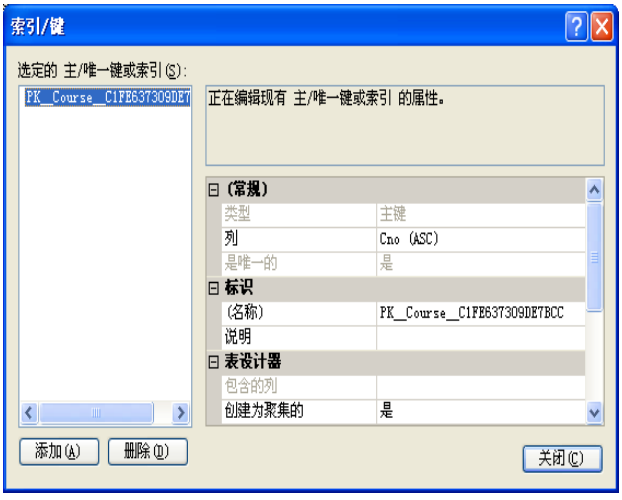


图 6-10　【索引/键】对话框

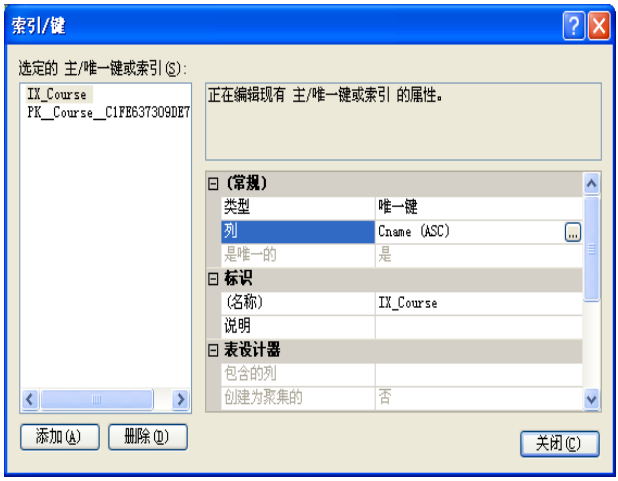


图 6-11　定义 UNIQUE 约束

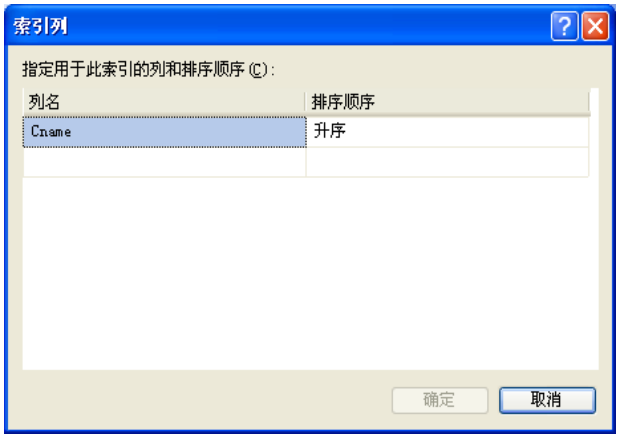


图 6-12　【索引列】对话框

使用 SQL Server 管理控制台删除 UNIQUE 约束的具体步骤如下。

（1）在【对象资源管理器】中，右击具有唯一约束的表，然后单击【设计】命令。此时，将在表设计器中打开该表。

（2）在【表设计器】菜单中单击【索引/键】命令。

（3）在【索引/键】对话框中，从【选定的主键/唯一键或索引】列表中选择唯一键。

（4）单击【删除】按钮，如图 6-13 所示。在保存表时，系统将从数据库中移除该约束。

2）使用 T-SQL 语句定义 UNIQUE 约束

在 T-SQL 语句中，可以使用 CREATE TABLE 语句创建 UNIQUE 约束，语法格式如下：

```
CREATE TABLE table_name
(column_name datetype [CONSTRAINT constraint_name] NOT NULL UNIQUE
[CLUSTERED|NONCLUSTERED][,...n])
```

图 6-13　删除 UNIQUE 约束

其中，各参数的含义如下。

- table_name：指定的表名。
- column_name：指定的列名。
- CONSTRAINT：指定约束名 constraint_name。
- UNIQUE：定义约束类型。
- CLUSTERED|NONCLUSTERED：定义约束的索引类型。

【例 6.11】　下面的语句用于创建表【Course】，并把【Cname】列定义为 UNIQUE 约束。

```
USE db_stu
CREATE TABLE Course
(Cno char(5)PRIMARY KEY NOT NULL,
Cname char(30)UNIQUE NOT NULL,
Ccredit int NULL,
Chours int NULL
)
```

也可以使用 ALTER TABLE 语句为已存在的表创建 UNIQUE 约束，其语法格式如下：

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name]
UNIQUE CLUSTERED|NOTCLUSTERED(column[,...n])
```

【例 6.12】　下面的语句用于把表【Course】中的【Cname】列定义为 UNIQUE 约束。

```
USE db_stu
ALTER TABLE Course
ADD CONSTRAINT Cname
UNIQUE CONSTRAINT
GO
```

可以使用 DROP 语句删除 UNIQUE 约束，其语法格式如下：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name[,...n]
```

【例 6.13】　下面的语句用于删除已有的 PRIMARY KEY 约束。

```
USE db_stu
ALTER TABLE Course
DROP CONSTRAINT Cno
GO
```

6.3.4　参照完整性的实现

参照完整性的实现也是表与表之间的参照关系，定义参照关系也可以通过对象资源管理器和 T-SQL 语句来实现。

FOREIGN KEY 约束可以定义表与表之间的参照关系。创建表时，可以创建 FOREIGN KEY 约束作为表定义的一部分。如果表已经存在，则可以添加 FOREIGN KEY 约束（假设该 FOREIGN KEY 约束被链接到了另一个或同一个表中某个现有的 PRIMARY KEY 约束或 UNIQUE 约束）。一个表可含有多个 FOREIGN KEY 约束。FOREIGN KEY 约束也可以通过 SQL Server 管理控制台和 T-SQL 语句来实现。

1．使用 SQL Server 管理控制台

具体操作步骤如下。

（1）在对象资源管理器中，用鼠标右键单击将位于关系的外键方的表，再单击【设计】按钮。此时，将在表设计器中打开该表。

（2）在【表设计器】菜单上，单击【关系】命令，如图 6-14 所示。

（3）在【外键关系】对话框中，单击 添加(A) 按钮。【选定的关系】列表中将显示关系及系统提供的名称，格式为 FK_<tablename>_<tablename>，其中 tablename 是外键表的名称，如图 6-15 所示。

（4）在【选定的关系】列表中单击该关系。单击右侧网格中的【表和列规范】，再单击该属性右侧的省略号（…），如图 6-16 所示。

（5）在【表和列】对话框中，从【主键表】下拉列表中选择位于关系主键方的表，如图 6-17 所示。

（6）在下方的网格中，选择要分配给表的主键的列。在每列左侧的相邻网格单元格中，选择外键表的相应外键列。表设计器将为此关系提供一个建议名称。若要更改此名称，请编辑【关系名】文本框的内容，如图 6-18 所示。

（7）选择【确定】按钮即可创建关系。

（8）如果要修改关系，重新进行第（5）、（6）步的操作即可。

（9）如果要删除关系，在【外键关系】对话框中选中要删除的关系，而后单击 删除(D) 按钮即可，如图 6-19 所示。

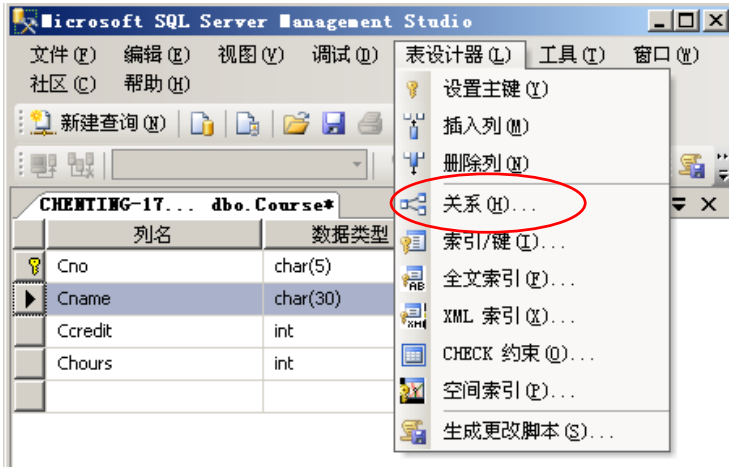


图 6-14　选择【关系】命令

图 6-15　【外键关系】对话框

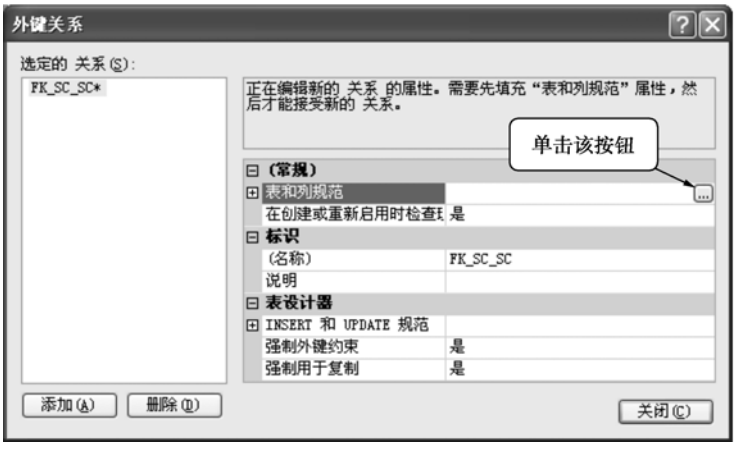


图 6-16　选定关系

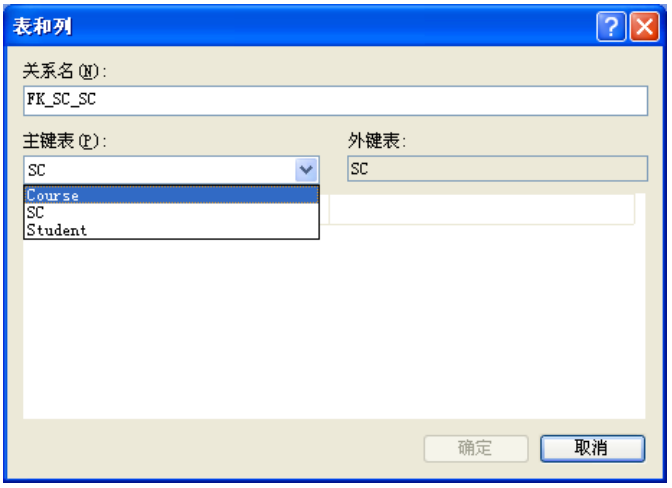


图 6-17　选择主键表

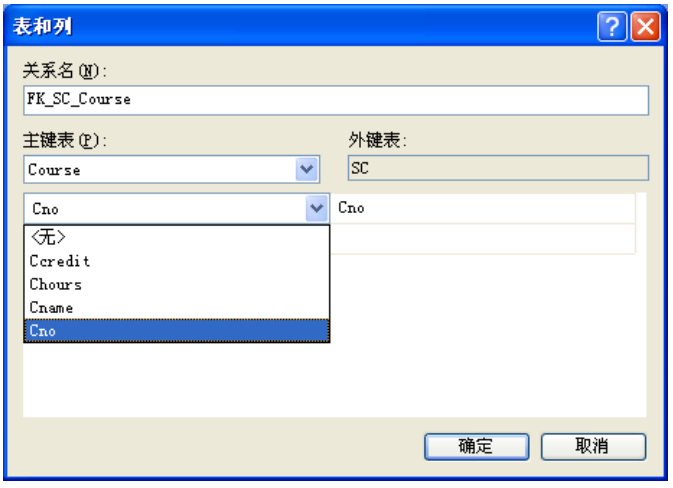


图 6-18　选择外键列

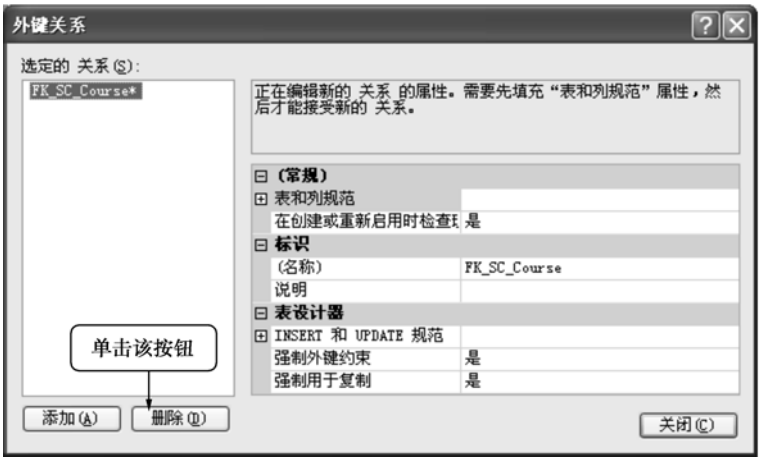


图 6-19　删除外键

2．使用 T-SQL 语句定义参照关系

在 T-SQL 语句中，可以使用 ALTER TABLE 语句定义已有表之间的参照关系，其语法格式如下：

```
ALTER TABLE table_name
ADD[CONSTRAINT constraint_name]FOREIGN KEY(column[,...n])REFERENCES ref_table(ref_column[,...n])
```

其中各参数的含义如下。

- table_name：指定被修改的从表名。
- column：指定表中外键的列名称。
- ref_table：指定主表的表名。
- ref_column：指定主表中主键的列名称。

【例 6.14】　下面的语句用于定义数据库【db_stu】中表【Course】和表【SC】的参照关系。

```
USE db_stu
ALTER TABLE SC
ADD CONSTRAINT Course
FOREIGN KEY (Cno) REFERENCES SC(Cno)
```

在 T-SQL 语句中，可以使用 CREATE TABLE 语句创建表，并定义参照关系，其语法格式如下：

```
CREATE TABLE table_name
 (column_name datatype[FOREIGN KEY] REFERENCES ref_table(ref_column)[,...n])
```

也可以使用 DROP 语句来删除表之间的参照关系，其语法格式如下：

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name
```

【例 6.15】　下面的语句用于删除表【SC】中【Cno】列的外键约束。

```
USE db_stu
ALTER TABLE SC
DROP CONSTRAINT Cno
```

习　　题

1．SQL Server 2008 中用来强制列中数据的完整性的机制有哪些？

2．在表【Student】中，字段【Sname】中的每个记录要求唯一，请使用 SQL Server 管理控制台为字段【Sname】定义 UNIQUE 约束。

3．试述索引的概念与作用。

4．请使用 SQL Server 管理控制台创建表【Student】和表【SC】之间的参照关系。

第 7 章　存储过程和触发器

- 了解存储过程、触发器的概念
- 掌握存储过程、触发器的创建、修改和删除方法

7.1　存 储 过 程

存储过程是一组存储在数据库中的 SQL 子程序，用户可以直接调用执行相同操作的存储过程来保证数据的一致性。

7.1.1　存储过程的类型

在 Microsoft SQL Server 中有 3 种可用的存储过程，分别为用户定义的存储过程、扩展存储过程和系统存储过程。

1．用户定义的存储过程

存储过程是指封装了可重用代码的模块或例程。存储过程可以接受输入参数、向客户端返回表格或标量结果和消息、调用数据定义语言（DDL）和数据操作语言（DML）语句，然后返回输出参数。在 SQL Server 2008 中，存储过程有两种类型，即 TransacT-SQL 和 CLR。

1）TransacT-SQL

TransacT-SQL 存储过程是指保存的 TransacT-SQL 语句集合，它可以接受和返回用户提供的参数。例如，存储过程中可能包含根据客户端应用程序提供的信息在一个或多个表中插入新行所需的语句。存储过程也可能从数据库向客户端应用程序返回数据。例如，电子商务 Web 应用程序可能使用存储过程根据联机用户指定的搜索条件返回有关特定产品的信息。

2）CLR

CLR 存储过程是指对 Microsoft .NET Framework 公共语言运行时方法的引用，它可以接受和返回用户提供的参数。它们在 .NET Framework 程序集中是作为类的公共静态方法实现的。

2．扩展存储过程

扩展存储过程允许用户使用编程语言（如 C）创建自己的外部例程。扩展存储过程是指 Microsoft SQL Server 的实例可以动态加载和运行的 DLL。扩展存储过程直接在 SQL Server 实例的地址空间中运行，可以使用 SQL Server 扩展存储过程 API 完成编程。

3．系统存储过程

它是由 SQL server 系统提供的存储过程，定义在系统数据库中，并以 sp_开头，存储过程允许具有执行系统存储过程权限的用户执行修改表的任务，并且可以在所有数据库中执行。

7.1.2　用户存储过程的创建与执行

1．用户存储过程的创建

在 SQL Server 2008 中用户只能在当前数据库中创建存储过程，并且过程名称不应该与其他过程名称相同。创建存储过程有使用 T-SQL 语句和使用对象资源管理器两种方法。

1）用 T-SQL 语句创建存储过程

在 T-SQL 语句中，用户可以使用 CREATE PROCEDURE 来创建存储过程，其语法格式如下：

```
CREATE  PROCEDURE procedure_name
[WITH {RECOMPILE|ENCRYPTION|RECOMPILE},ENCRYPTION]
AS
Sql_statement[,…n]
```

其中，各参数的含义如下。

- procedure_name：指定存储过程的名称，其名称必须符合标志符命名规则，且在数据库中必须唯一。
- WITH 子句：指定设置选项。
- RECOMPILE：表明 SQL Server 在每次运行存储过程时重新编译。
- ENCRYPTION：表明 SQL Server 加密存储过程的文本。
- Sql_statement：在存储过程中包含的 T-SQL 语句。
- n：表示此过程可以包含多条 T-SQL 语句。

【例 7.1】　下面创建从【db_stu】数据库中查询选课表【SC】中学生成绩信息的存储过程，其具体操作如下。

（1）在查询分析器中输入如下语句：

```
USE db_stu
GO
CREATE PROCEDURE STU
AS
SELECT Student.Sno,Cno,Grade
FROM Student,SC
WHERE Student.Sno=SC.Sno
GO
```

（2）在工具栏中单击 ！执行(X) 按钮，完成存储过程的创建，此时在【db_stu】数据库的存储过程窗口可以看到新建的 STU 的存储过程，如图 7-1 所示。

2）使用对象资源管理器创建存储过程

在对象资源管理器下创建存储过程的具体操作步骤如下。

（1）在【对象资源管理器】中，连接到某个数据库引擎实例，再展开该实例。

（2）展开【数据库】→【db_stu】→【可编程性】。

（3）右击【存储过程】，再单击【新建存储过程】命令，如图 7-2 所示。

（4）在【查询】菜单上，单击【指定模板参数的值】命令，如图 7-3 所示。

（5）在【指定模板参数的值】对话框中，【值】列包含参数的建议值。接受这些值或将其替换为新值，再单击【确定】按钮，如图 7-4 所示。

（6）在查询编辑器中，使用过程语句替换 SELECT 语句，如图 7-5 所示。

（7）若要测试语法，请在【查询】菜单上单击【分析】命令。

（8）若要创建存储过程，请在【查询】菜单上单击【执行】命令，如图 7-6 所示。

（9）若要保存脚本，请在【文件】菜单上单击【保存】命令。接受该文件名或将其替换为新的名称，再单击“保存”按钮。

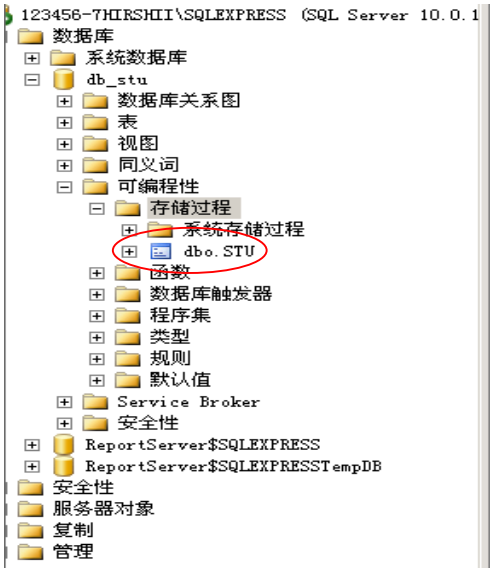


图 7-1　T-SQL 语句创建存储过程

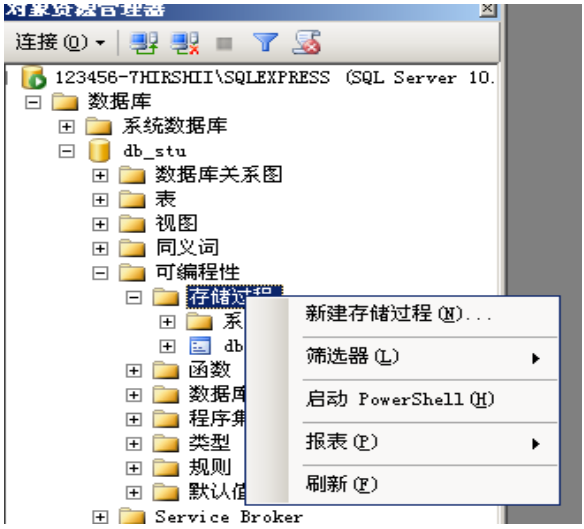


图 7-2　新建存储过程

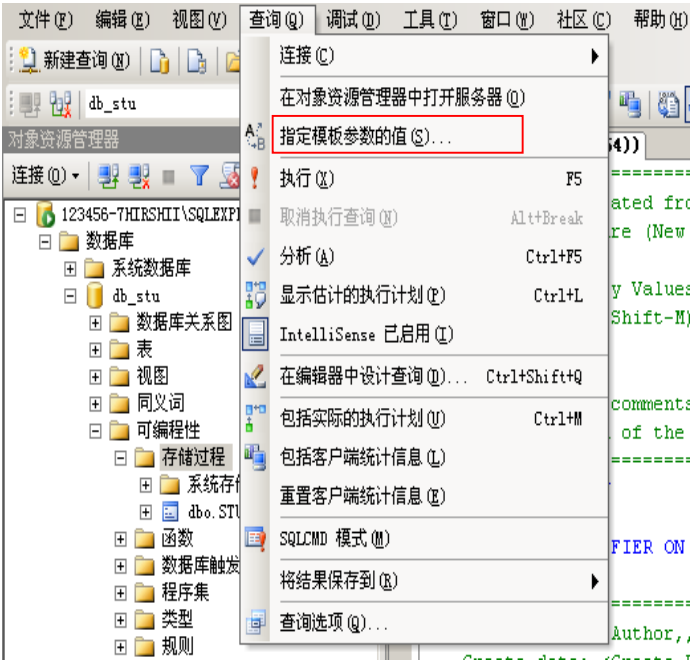


图 7-3　打开参数对话框

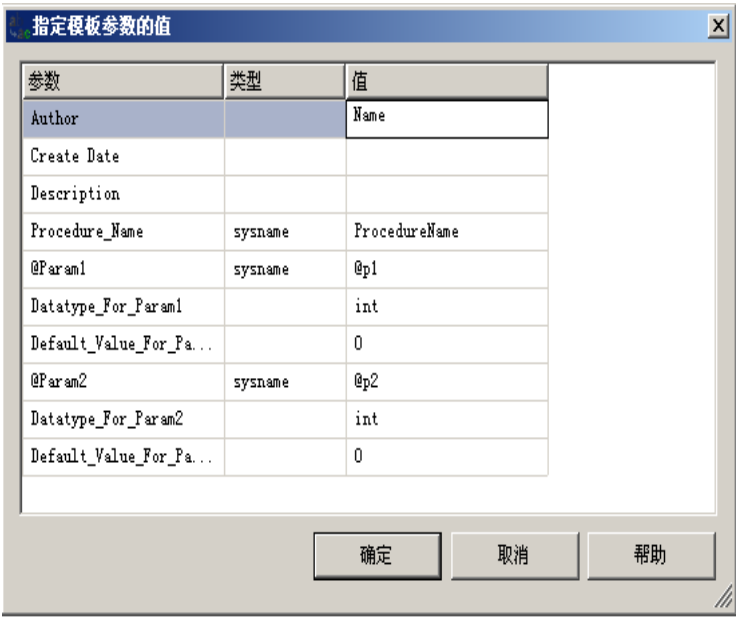


图 7-4　参数对话框

```
SQLQuery1.s...rator (54))*
-- Author:      Name
-- Create date:
-- Description:
-- ============================================
CREATE PROCEDURE ProcedureName
    -- Add the parameters for the stored procedure here
    @p1 int = 0,
    @p2 int = 0
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT Student.Sno,Cno,Grade
FROM Student,SC
WHERE Student.Sno=SC.Sno
END
GO
```

图 7-5　替换语句

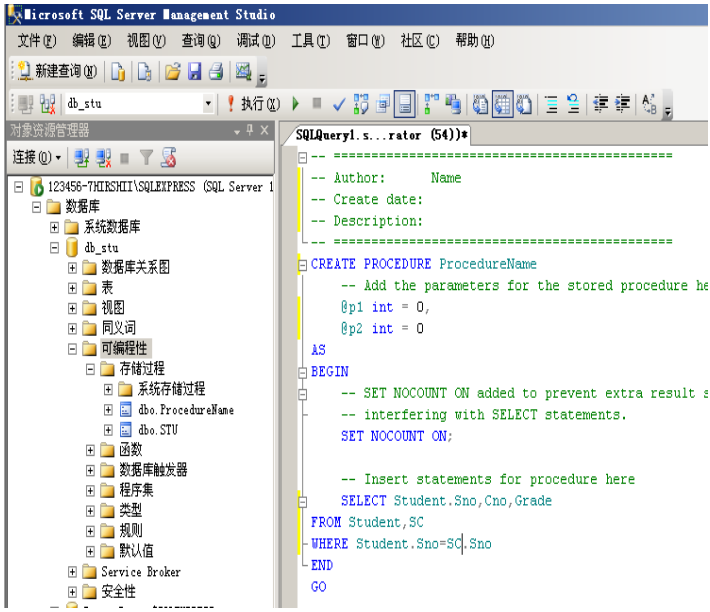


图 7-6　创建存储过程

2．用户存储过程的执行

已创建的存储过程可以使用 EXECUTE（或简写 EXEC）命令执行，下面调用执行已创建的存储过程 STU。其具体操作如下。

（1）在查询分析器中输入如下语句：

```
USE db_stu
GO
EXEC STU
```

（2）在工具栏中单击 ! 执行(X) 按钮，执行存储过程。

另外，可以在将要执行的存储过程的名字上单击鼠标右键，选择【执行存储过程】命令，

如图 7-7 所示。执行结果如图 7-8 所示。

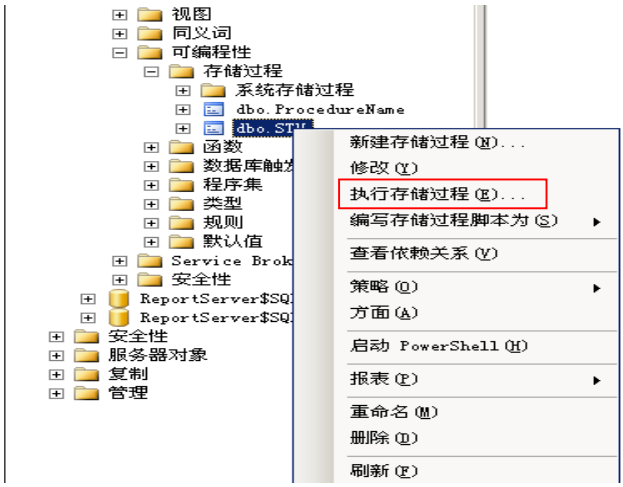


图 7-7　【执行存储过程】命令

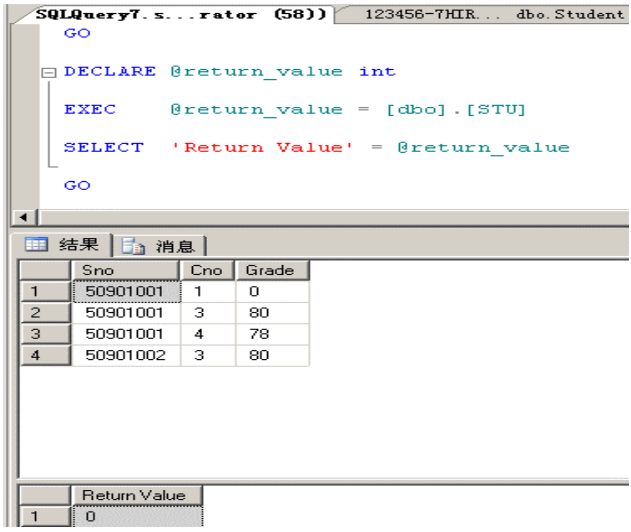


图 7-8　存储过程执行结果

7.1.3　用户存储过程的编辑修改

创建存储过程后，如果需要改变存储过程中的语句或参数，可以修改或重命名存储过程。编辑修改存储过程的步骤如下。

（1）在【对象资源管理器】中，连接到某个数据库引擎实例，再展开该实例。

（2）依次展开【数据库】，以及存储过程所属的数据库。例如，展开【数据库】→【db_stu】→【可编程性】。

（3）展开【存储过程】，用鼠标右键单击要修改的过程，再单击【修改】命令，如图 7-9 所示。

（4）利用 T-SQL 语句中的 ALTER PROCEDURE 命令修改存储过程的文本，如图 7-10 所示。

（5）若要测试语法，请在【查询】菜单上单击【分析】命令。

（6）若要修改存储过程，请在【查询】菜单上单击【执行】命令，结果如图 7-11 所示。

（7）若要保存脚本，请在【文件】菜单上单击【另存为】命令，接受文件名或使用新名。

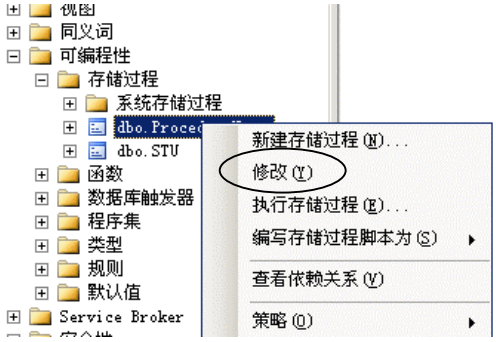


图 7-9　选中【修改】命令

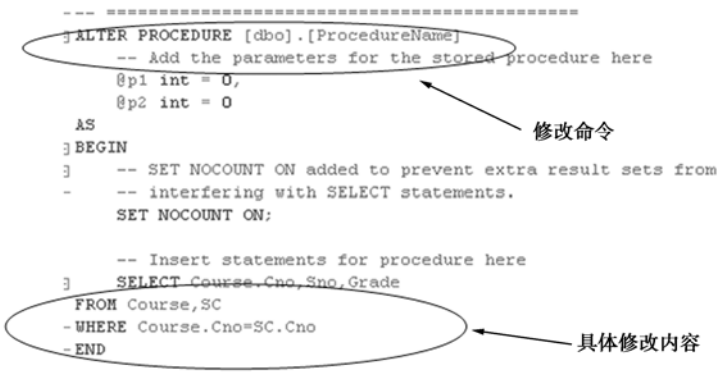
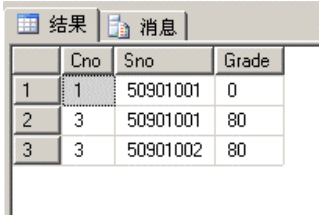


图 7-10　修改存储过程



	Cno	Sno	Grade
1	1	50901001	0
2	3	50901001	80
3	3	50901002	80

图 7-11　修改结果

7.1.4　用户存储过程的删除

当存储过程不再使用时，可以将其删除，存储过程的删除也有使用 T-SQL 语句和使用对象资源管理器两种方法。

1．使用 T-SQL 语句删除存储过程

在 T-SQL 语句中，可以使用 DROP PROCEDURE 语句删除存储过程，其语法格式如下：

```
DROP PROCEDURE {procedure_name}[,..n]
```

【例 7.2】　下面的语句可用于删除存储过程 procedurename。

```
USE db_stu
DROP PROCEDURE procedurename
```

2．使用对象资源管理器删除存储过程

（1）在【对象资源管理器】中，连接到某个数据库引擎实例，再展开该实例。

（2）依次展开【数据库】→【db_stu】→【可编程性】。

（3）展开【存储过程】，右击要删除的过程，再单击【删除】命令。

（4）若要查看基于存储过程的对象，请单击【显示依赖关系】命令。

（5）确认已选择了正确的存储过程，再单击【确定】按钮，如图 7-12 所示。

（6）从依赖对象和脚本中删除存储过程名称。

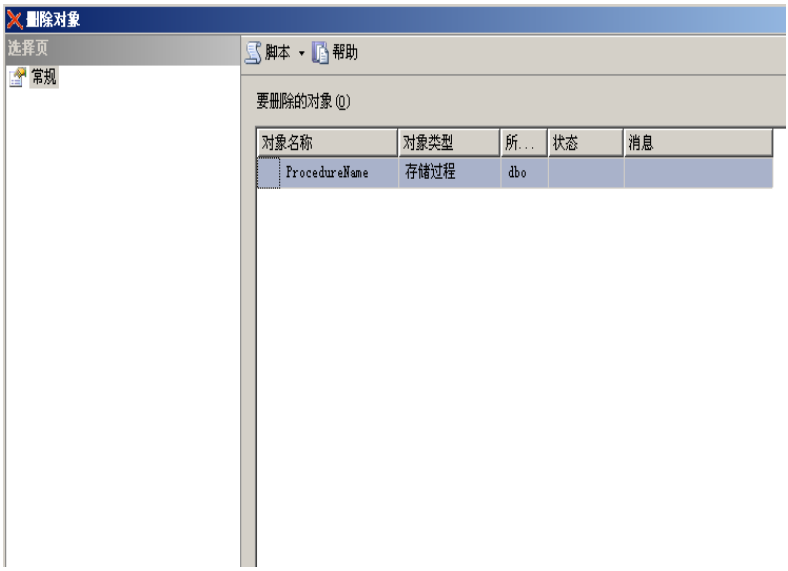


图 7-12　【删除对象】对话框

7.2　触　发　器

触发器是一种特殊的存储过程，用于改变指定表中的数据，它能自动执行并成为 SQL 语句的一部分。触发器创建在一个表的基础上，并和一个或多个数据的编辑操作相关联。在 SQL Server 2008 中触发器分为两类：DML 触发器和 DDL 触发器。其中 DML 触发器是当数据库服务器中发生数据操作语言事件时执行的存储过程。DML 触发器又分为两大类：AFTER 触发器和 INSTEAD OF 触发器。而 DDL 触发器是在响应数据定义语言事件时执行的存储过程。

7.2.1　利用 SQL 语句创建触发器

1．用 CREATE TRIGGER 命令创建 AFTER 触发器

其语法格式如下：

```
CREATE TRIGGER trigger_name
```

```
ON [table_name|view_name] AFTER [INSERT][,][UPDATE][,][DELETE]
AS
Sql_statement[,…n]
```

其中，各参数的含义如下。

- trigger_name：触发器的名字，在数据库中必须唯一。
- Sql_statement：SQL 语句。

【例 7.3】　在表【Student】中创建一个名字为 studel 的触发器，当要删除学生时，触发器被触发，检查表【SC】，并同时删除相同学号的学生。打开查询分析器，在里面输入以下代码，结果如图 7-13 所示。

```
USE db_stu
GO
CREATE TRIGGER studel
ON Student
AFTER　DELETE
AS
BEGIN
DELETE FROM SC
where Sno IN (select Sno from deleted)
END
```

图 7-13　创建 AFTER 触发器

2．用 CREATE TRIGGER 命令创建 INSTEAD OF 触发器

其语法格式如下：

```
CREATE TRIGGER trigger_name
ON [table_name|view_name] INSTEAD OF[INSERT][,][UPDATE][,][DELETE]
AS
Sql_statement[,…n]
```

其中各参数的含义与前面相同。

【例 7.4】　在表【Student】中创建一个名字为 stuinsert 的触发器，当要插入新学生时，触发器被触发，检查表【SC】，并同时插入相同学号的学生。

```
CREATE TRIGGER stuinsert
ON Student
INSTEAD OF INSERT
AS
INSERT　INTO Student
VALUES('50901006','zhoulan','F','17','CS')
INSERT　INTO SC
VALUES('50901006',null,null,null,null')
```

7.2.2　利用 SQL Server Management Studio 创建触发器

在 SQL Server Management Studio 中创建触发器的具体步骤如下。

（1）在【对象资源管理器】中，登录服务器，并展开【数据库】→【db_stu】→【表】，在展开项中选择表【Student】，单击它前面的加号，在其展开项中选择“触发器”，右击，在弹出菜单中选择【新建触发器】命令，如图 7-14 所示。

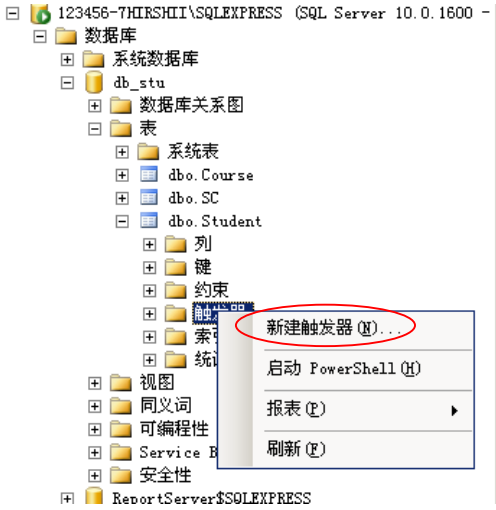


图 7-14　选择【新建触发器】命令

（2）在出现的窗口中输入 T-SQL 语句，如图 7-15 所示。

```
-- ============================================
CREATE TRIGGER <Schema_Name, sysname, Schema_Name>.<Trigger_Name, sysname, Trigger_Name>
    ON  <Schema_Name, sysname, Schema_Name>.<Table_Name, sysname, Table_Name>
    AFTER <Data_Modification_Statements, , INSERT,DELETE,UPDATE>
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for trigger here

END
GO
```

图 7-15　输入 T-SQL 语句

（3）输入完成后，单击菜单栏上的 执行(X) 按钮，在该表的【触发器】下面就可以看到新建的触发器了。

7.2.3　触发器的修改和删除

触发器建立好之后，根据具体的需要会对所建的触发器进行修改，如果不需要也可删除。

1．触发器的修改

修改触发器的步骤如下。

（1）展开数据库，找到【表】，在其中一个表的【触发器】项中选择需要修改的触发器的

名字。

（2）单击鼠标右键，在弹出的菜单中选择【修改】命令，在窗口右侧可以查看所选触发器的相关内容，如图 7-16 所示。

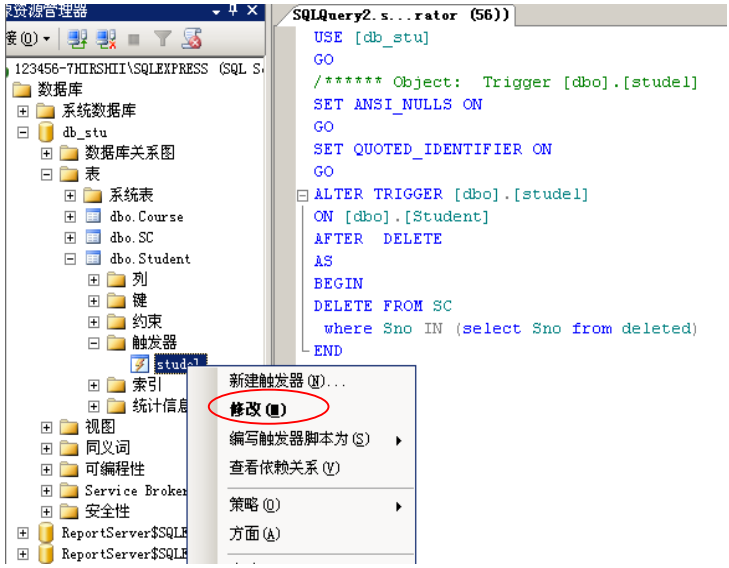


图 7-16　修改触发器

使用 ALTER TRIGGER 命令修改触发器的方法如下。

ALTER TRIGGER 命令的语法格式如下：

```
ALTER TRIGGER trigger_name
ON [table_name|view_name]    FOR [INSERT][,][UPDATE][,][DELETE]
AS
Sql_statement[,…n]
```

【例 7.5】　下面的语句用于修改数据库【db_stu】中的触发器 studel。

```
USE [db_stu]
GO
/****** Object:    Trigger [dbo].[studel]        Script Date: 10/12/2009 16:09:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER TRIGGER [dbo].[studel]
ON Course
AFTER    DELETE
AS
BEGIN
DELETE FROM SC
```

```
    where Cno IN (select Cno from deleted)
END
```

2．触发器的删除

当触发器不再需要时，应该将其删除。删除触发器时，其所在的表及表中的数据不受影响。如果删除表，则表中所有的触发器将被自动删除。

删除触发器可以使用对象资源管理器和 DROP TRIGGER 语句。

使用对象资源管理器删除触发器和修改触发器相似，在【触发器】项中选中要删除的触发器，单击鼠标右键，在弹出的菜单中选择【删除】命令，即可删除触发器，如图 7-17 所示。

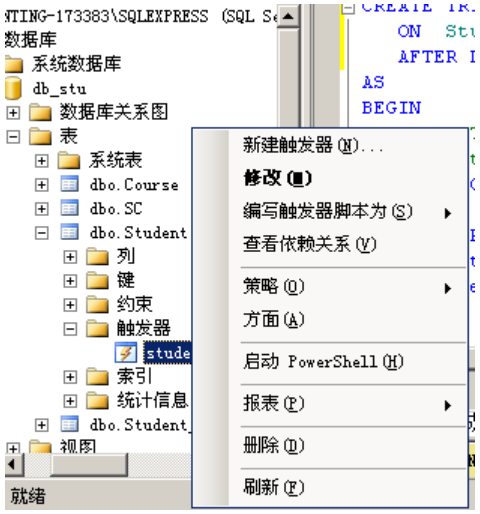


图 7-17　删除触发器

DROP TRIGGER 语句的语法格式如下：

```
DROP TRIGGER {trigger}[,…n]
```

【例 7.6】　下面的语句用于删除触发器 studel。

```
USE db_stu
DROP TRIGGER studel
GO
```

习　　题

1．如何执行存储过程？存储过程的执行有哪些特点？

2．存储过程与触发器的区别是什么？

3．怎样创建触发器？删除触发器后会不会影响表里的数据？

4．在【db_stu】数据库中创建一个名为【Course_ins】的触发器，当要插入新的课程时，触发器被触发，然后使用触发语句来取代这个插入操作。

5．参照本章所学知识，为数据库【db_stu】创建一个名为【spsel】的触发器，用于查询某位学生的所有信息。

第 8 章　备份恢复与导入/导出

- 了解备份和恢复的基本概念
- 掌握备份和恢复数据库的操作
- 熟悉备份和恢复的应用
- 了解导入和导出的基本概念
- 了解 BCP 实用程序
- 了解 DTS 的基本概念
- 掌握 DTS 导入/导出的使用方法

8.1　备份和恢复概述

8.1.1　备份和恢复需求分析

为了成功地设计备份和恢复策略，用户必须进行以下的实用性需求分析。

（1）备份需求：数据库停止使用能对用户造成多大的经济损失；如果遇到存储介质故障，会停工多长时间。保留数据是否重要。

（2）备份内容：数据库需备份的内容可分为系统数据库和用户数据库，系统数据库记录了重要的系统信息，用户数据库记录了用户的数据。

（3）备份人员：是否需要专业的系统管理员或数据库管理员。

（4）备份时间：当系统数据库 master、msdb 和 model 中任何一个被修改时，都需要进行备份；当用户数据库被创建、修改、建立索引时也需要进行备份。

（5）备份介质：备份介质是指数据库备份到的目标载体，如硬盘、磁盘。

（6）备份频率：指相隔多长时间备份一次，确定备份频率主要有两点，一是系统恢复的工作量，二是系统执行的事务量。

（7）备份限制：在备份过程中不允许使用 CREATE DATABASE、ALTER DATABASE、创建索引和日志命令。

（8）性能考虑：备份时选择合适的磁盘可以提高备份速度；备份到多个物理设备比一个要快；备份时应减少并发活动来提高备份速度。

8.1.2　数据库备份和恢复的基本概念

备份就是创建 SQL Server 数据库或事务日志的副本，数据库备份记录了在进行备份这一操作时数据库中所有数据的状态，以便在数据库遭到破坏时能够及时地将其恢复。

在 SQL Server 2008 中，数据备份的方法如下。

1．数据备份

（1）数据库备份：备份类型分别为数据库备份（整个数据库的完整备份）和差异数据库备份（数据库中所有文件的备份。此备份只包含自每个文件的最新数据库备份之后发生了修改的数据区）。

（2）部分备份：备份类型分别为部分备份（备份主文件组、所有读/写文件组，以及所有指定的只读文件或文件组中的所有完整数据。只读数据库的部分备份仅包含主文件组）和部分差异备份（仅包含自同一组文件组的最新部分备份以来发生了修改的数据区）。

（3）文件备份：备份类型分别为文件备份（一个或多个文件或文件组中所有数据的完整备份）和差异文件备份（一个或多个文件的备份，包含自每个文件的最新完整备份之后发生了更改的数据区）。

2．事务日志备份

在完整恢复模式或大容量日志恢复模式下，需要定期进行“事务日志备份”（或“日志备份”）。每个日志备份都包括创建备份时处于活动状态的部分事务日志，以及先前日志备份中未备份的所有日志记录。不间断的日志备份序列包含数据库的完整（连续不断的）日志链。在完整恢复模式下（或者在大容量日志恢复模式下的某些时候），连续不断的日志链使用户可以将数据库还原到任意时间点。

在创建第一个日志备份之前，必须先创建一个完整备份（如数据库备份）。因此，定期备份事务日志十分有必要，这不仅可以使工作丢失的可能性降到最低，而且还能截断事务日志。

恢复是指数据库备份后，一旦系统发生崩溃或者执行了错误的数据库操作，就可以从备份文件中恢复数据库。数据库恢复是指将数据库备份加载到系统中的过程。系统在恢复数据库的过程中，会自动执行安全性检查、重建数据库结构及完整的数据库内容。

SQL Server 2008 有 3 种恢复模式：简单恢复模式、完整恢复模式和大容量日志恢复模式。通常，数据库使用完整恢复模式或简单恢复模式。

用于还原和恢复数据的数据副本称为“备份”。使用备份可以在发生故障后还原数据。通过妥善的备份，可以从多种故障中恢复，如媒体故障、用户错误（误删除了某个表）、硬件故障（磁盘驱动器损坏或服务器报废）和自然灾难等。

8.2　备份操作和备份命令

8.2.1　创建备份设备

在进行备份以前首先必须指定或创建备份设备，备份设备是用来存储数据库、事务日志或文件和文件组备份的存储介质，备份设备可以是硬盘、磁带或管道。当使用磁盘时，SQL Server 2008 允许将本地主机硬盘和远程主机上的硬盘作为备份设备，备份设备在硬盘中是以文件的方式存储的。

创建备份设备有两种方法。

1．使用 SQL Server 对象资源管理器创建备份设备

（1）使用管理员账号登录到服务器，启动【对象资源管理器】。

（2）在【对象资源管理器】中，展开【服务器对象】→【备份设备】选项，单击鼠标右键，在弹出的快捷菜单中选择【新建备份设备】命令，如图 8-1 所示。

（3）在打开的【备份设备】对话框的【设备名称】文本框中输入新设备名称，如“db_stubackup”，单击 ... 按钮选择【文件】，如图 8-2 所示。

（4）单击 确定 按钮，完成新的备份设备的创建，如图 8-3 所示。

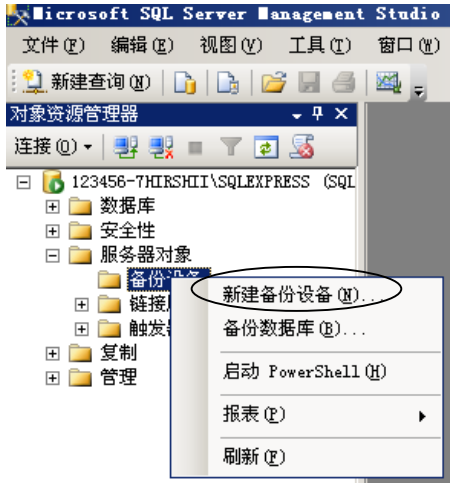


图 8-1　选择【新建备份设备】命令

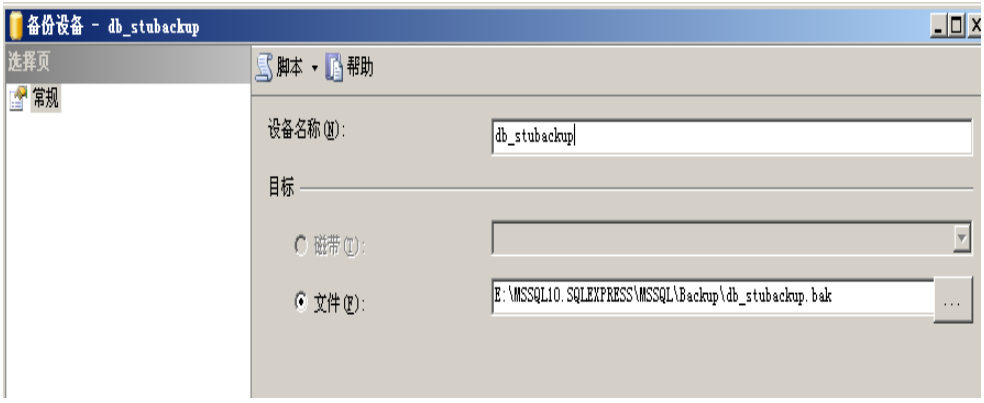


图 8-2　【备份设备】对话框

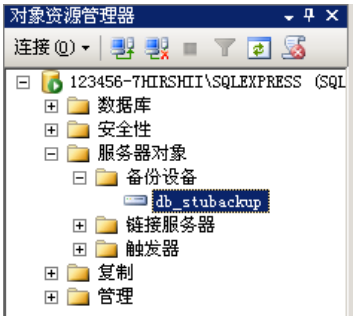


图 8-3　新建备份设备

2．使用 T-SQL 语句创建备份设备

在 SQL Server 中，可以使用 sp_addumpdevice 语句创建备份设备，其语法形式如下：

```
sp_addumpdevice
{'device_type'}[,'logical_name'][,'physical_name'][,{{controller_type|'device_status'}}]
```

其中，device_type 指介质类型，可以是{DISK|TAPE|PIPE}；logical_name 和 physical_name 分别为制定的逻辑名和物理名。

【例 8.1】　下面的语句用于在本地磁盘中创建一个备份设备，如图 8-4 所示。打开查询分析器，在里面输入如下语句：

```
USE db_stu
exec sp_addumpdevice 'disk','stu','d:\data\backup\stu'
```

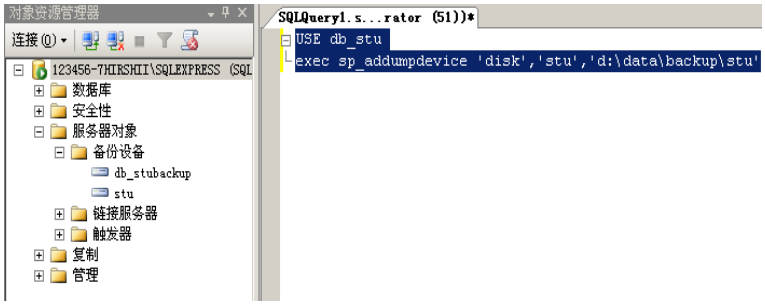


图 8-4　T-SQL 语句创建备份设备

8.2.2　使用对象资源管理器进行数据库备份

SQL Server 2008 可以为数据库进行备份，以减少数据库管理员的工作负担。其使用方法如下。

（1）使用管理员账号登录到服务器，启动【对象资源管理器】。

（2）在【对象资源管理器】中，展开【服务器对象】→【备份设备】选项，单击鼠标右键，在弹出的快捷菜单中选择【备份数据库】命令，如图 8-5 所示。

（3）在弹出的【备份数据库】对话框中，可以选择要备份的数据库、备份的类型、备份集过期时间，以及备份的存储路径，如图 8-6 所示。

（4）单击 确定 按钮，备份成功，如图 8-7 所示。

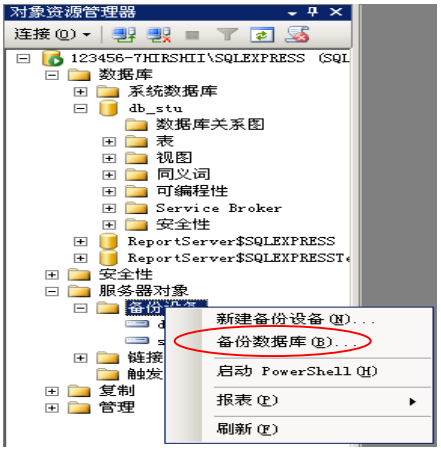


图 8-5　选择【备份数据库】命令

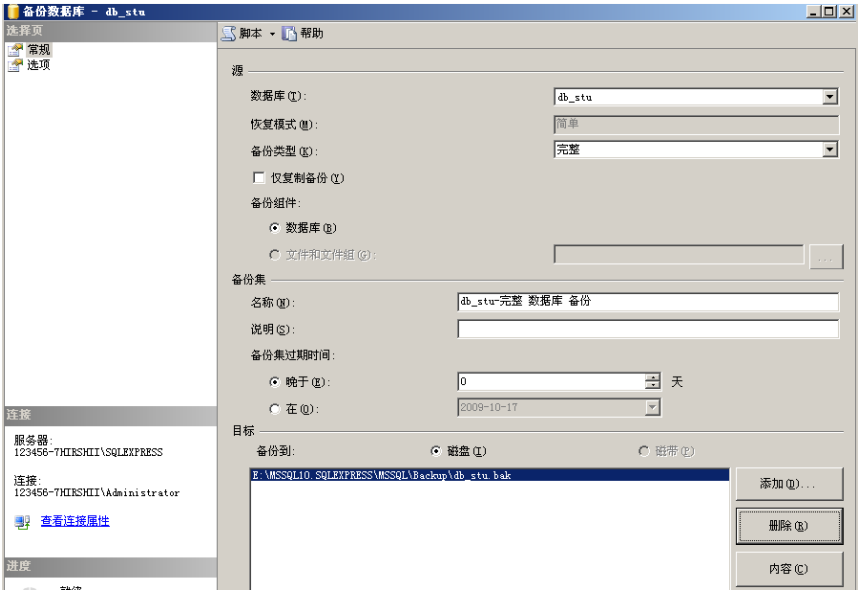


图 8-6　【备份数据库】对话框

图 8-7　备份成功

8.3　恢复操作和恢复命令

恢复是与备份相对应的操作，其目的在于当系统出现问题时可以利用备份将数据库恢复到正常的状态。

8.3.1　检查点（check point）

检查点将脏数据页从当前数据库缓冲区的高速缓存中刷新到磁盘上。这最大限度地减少了数据库完整恢复时必须处理的活动日志部分。当执行以下操作时将发生检查点的操作。

（1）显式执行 CHECKPOINT 语句，则用于连接的当前数据库中出现检查点。

（2）在数据库中执行了最小日志记录操作。例如，在使用大容量日志恢复模式的数据库中执行大容量复制操作。

（3）已经使用 ALTER DATABASE 添加或删除了数据库文件。

（4）通过 SHUTDOWN 语句或通过停止 SQL Server（MSSQLSERVER）服务停止了 SQL Server 实例。任一操作都会在 SQL Server 实例的每个数据库中形成一个检查点。

（5）SQL Server 实例在每个数据库内定期生成自动检查点，以减少实例恢复数据库所需的时间。

（6）进行了数据库备份。

（7）执行了需要关闭数据库的活动。例如，将 AUTO_CLOSE 设置为 ON 并且关闭了数据

库的最后一个用户连接，或者执行了需要重新启动数据库的数据库选项更改。

8.3.2　数据库的恢复命令

在 T-SQL 语句中，可以使用 RESTORE 语句来恢复数据库。

1．恢复整个数据库

语法格式如下：

```
RESTORE DATABASE data_name FROM back_device [WITH[FILE=n][,NORECOVERY|RECOVERY]
[,REPLACE]]
```

其中，各参数的含义如下。

- FILE=n：指从设备上的第 n 个备份中恢复。
- RECOVERY：指定在数据库恢复完成后，SQL Server 回滚被恢复的数据库中所有未完成的事务，以保持数据库的一致性。
- REPLACE：表示如果恢复的数据库名称与已有的某数据库重名，则首先删除原来的数据库，然后重建。

2．恢复事务日志

语法格式如下：

```
RESTORE DATABASE data_name FROM back_device [WITH[FILE=n][,NORECOVERY|RECOVERY]
```

恢复事务日志必须在进行完全数据库恢复后才能进行。

3．恢复特定的文件或文件组

语法格式如下：

```
RESTORE DATABASE data_name FLIE=file_name|FILEGROUP=groupfile_name FROM back_device
[WITH[FILE=n][,NORECOVERY|RECOVERY]
```

8.3.3　使用对象资源管理器恢复数据库

下面讲解如何使用对象资源管理器恢复数据库。具体步骤如下。

（1）连接到相应的 Microsoft SQL Server 数据库引擎实例之后，在对象资源管理器中，单击服务器名称以展开服务器树。

（2）展开【数据库】，用鼠标右键单击【系统数据库】，在弹出的菜单中选择【还原数据库】命令，如图 8-8 所示。

（3）打开【还原数据库】对话框，如图 8-9 所示。

（4）在【常规】页上，还原数据库的名称将显示在【目标数据库】列表框中。若要创建新数据库，请在列表框中输入数据库名。

（5）在【目标时间点】文本框中，可以保留默认值（【最近状态】），也可以单击【浏览】按钮打开【时点还原】对话框，以选择具体的日期和时间。

（6）若要指定要还原的备份集的源和位置，请单击以下选项之一。

- 【源数据库】：在列表框中输入数据库名称。
- 【源设备】：单击【浏览】按钮，打开【指定备份】对话框。在【备份媒体】列表框中，从列出的设备类型中选择一种。若要为【备份位置】列表框选择一个或多个设备，

请单击【添加】按钮。将所需设备添加到【备份位置】列表框后，单击【确定】按钮返回到【常规】页，如图 8-10 所示。

（7）在【选择用于还原的备份集】网格中，选择用于还原的备份。此网格将显示对于指定位置可用的备份。默认情况下，系统会推荐一个恢复计划。若要覆盖建议的恢复计划，可以更改网格中的选择。

（8）单击 确定 按钮，即可完成数据库的还原。

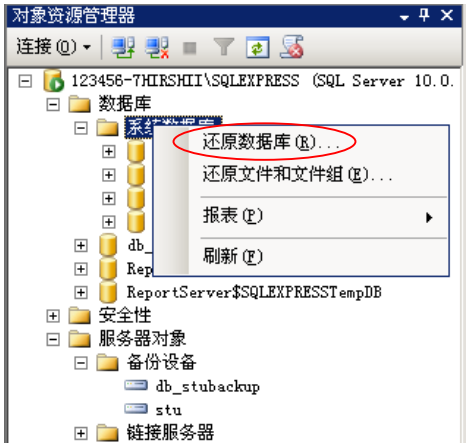


图 8-8　选择【还原数据库】命令

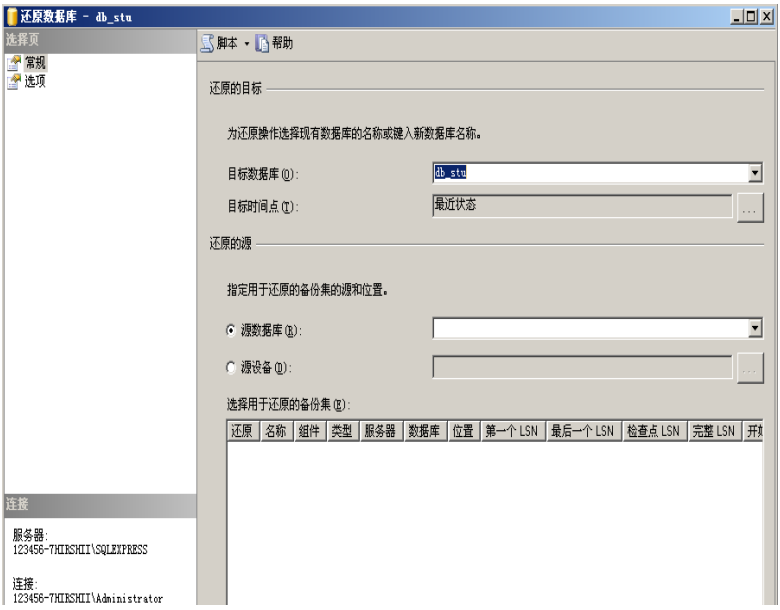


图 8-9　【还原数据库】对话框

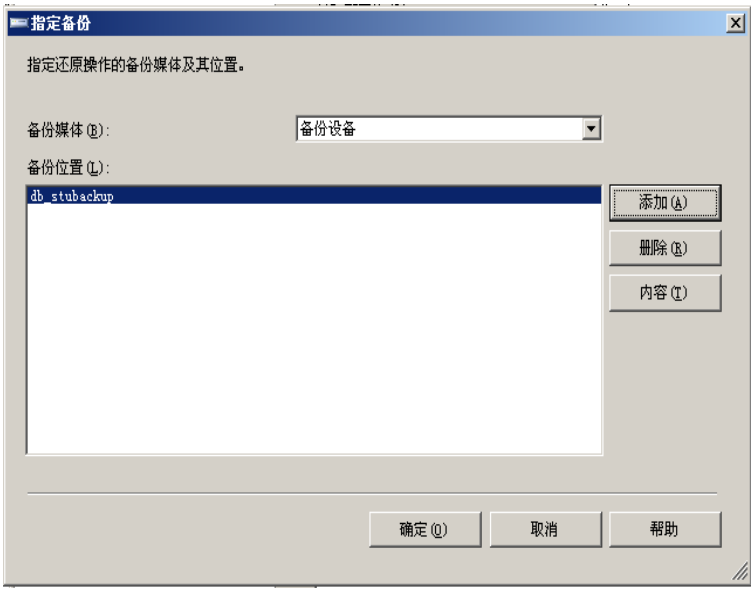


图 8-10　指定还原的源设备

8.4　导入/导出

由于应用环境的多变，在不同的环境下可能需要不同的数据格式，这就需要在各种格式之间进行转换，有时也需要将其他数据导入到 SQL Server 中。

8.4.1　导入/导出概念

导入数据是从外部程序（如文本文件、Excel 等）将数据引入到 SQL Server 表中；导出数据是从 SQL Server 数据库中引出数据到其他程序中，如将 SQL Server 数据表中的数据引出到 Microsoft Access 数据表中。

在 SQL Server 中，可以调用命令行工具 BCP 处理数据；也可以运用导入/导出向导来导入/导出数据。

8.4.2　使用 BCP 实用程序导入/导出数据

BCP 是 SQL Server 的一个批量复制实用程序，其功能是将数据库表中的数据复制到某个数据文件中，或把某个数据文件中的数据复制到数据表中，这种方法常用于 ASCII 文本文件与数据表进行交换。

BCP 实用程序的语法格式如下：

```
bcp {[[database_name.][schema].]{table_name | view_name} | "query"}
{in | out | queryout | format} data_file
[－m max_errors] [－f format_file] [－x] [－e err_file]
[－F first_row] [－L last_row] [－b batch_size]
[－n] [－c] [－N] [－w] [－V (70 | 80 | 90 )]
[－q] [－C { ACP | OEM | RAW | code_page } ] [－t field_term]
```

```
[－r row_term] [－i input_file] [－o output_file] [－a packet_size]
[－S server_name[\instance_name]] [－U login_id] [－P password]
[－T] [－v] [－R] [－k] [－E] [－h"hint [,…n]"]
```

其中，各参数的含义如下。

- database_name：指定的表或视图所在数据库的名称。如果未指定，则使用用户的默认数据库。
- owner：表或视图所有者的名称。如果执行该操作的用户拥有指定的表或视图，则 owner 是可选的。如果未指定 owner，并且执行该操作的用户不是指定的表或视图的所有者，则 SQL Server 将返回错误消息，而且该操作将取消。
- table_name：将数据导入 SQL Server 时为目标表名称，将数据从 SQL Server 导出时为源表名称。
- view_name：将数据复制到 SQL Server 时为目标视图名称，从 SQL Server 中复制数据时为源视图名称。只有其中所有列都引用同一个表的视图才能用做目标视图。
- " query "：一个返回结果集的 TransacT-SQL 查询。如果该查询返回多个结果集（如指定 COMPUTE 子句的 SELECT 语句），则只将第一个结果集复制到数据文件，而忽略后续的结果集。请将查询放在英文双引号中，将查询中嵌入的任何内容放在英文单引号中。从查询中大容量复制数据时，还必须指定 queryout。

详细参数说明请参阅 SQL Server 2008 的帮助文件。

【例 8.2】　下面的语句用于将【db_stu】数据库中的包【Student】导出到文本文件中，目标位置为 d:\data\stud.txt。

```
Bcp "db_stu..Student" out d:\data\stud.txt
```

8.4.3　使用导入/导出向导

使用导入/导出向导进行数据导入/导出操作的具体步骤如下。

（1）连接到相应的 Microsoft SQL Server 数据库引擎实例之后，在对象资源管理器中，单击服务器名称以展开服务器树。

（2）选定将要导入或导出数据的数据库，单击鼠标右键，在弹出的菜单中选择【任务】→【导入数据】命令，弹出【欢迎使用 SQL Server 导入和导出向导】对话框，如图 8-11 所示。

（3）单击 下一步(N) > 按钮，弹出如图 8-12 所示的对话框，选择数据源，再次单击 下一步(N) > 按钮。

（4）弹出【选择目标】对话框，如图 8-13 所示。单击 下一步(N) > 按钮。

（5）弹出【指定表复制或查询】对话框，如图 8-14 所示。单击 下一步(N) > 按钮。

（6）弹出【选择源表和源视图】对话框，如图 8-15 所示。选择源表，单击 下一步(N) > 按钮。

（7）弹出【运行包】对话框，如图 8-16 所示。单击 完成(F) >>| 按钮，弹出【执行成功】对话框，如图 8-17 所示。至此就完成了数据表的导入。

（8）刷新【对象资源管理器】，就会看到导入的数据表，如图 8-18 所示。

图 8-11　【欢迎使用 SQL Server 导入和导出向导】对话框

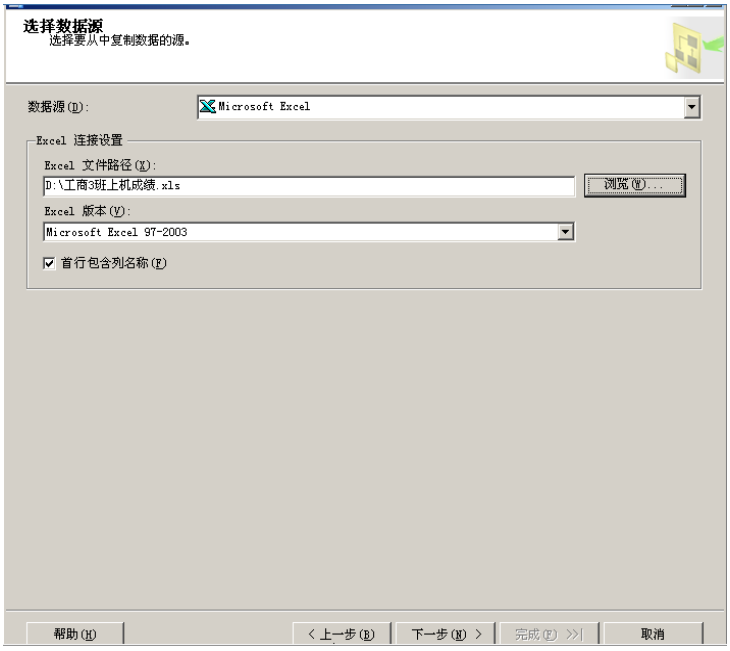


图 8-12　【选择数据源】对话框

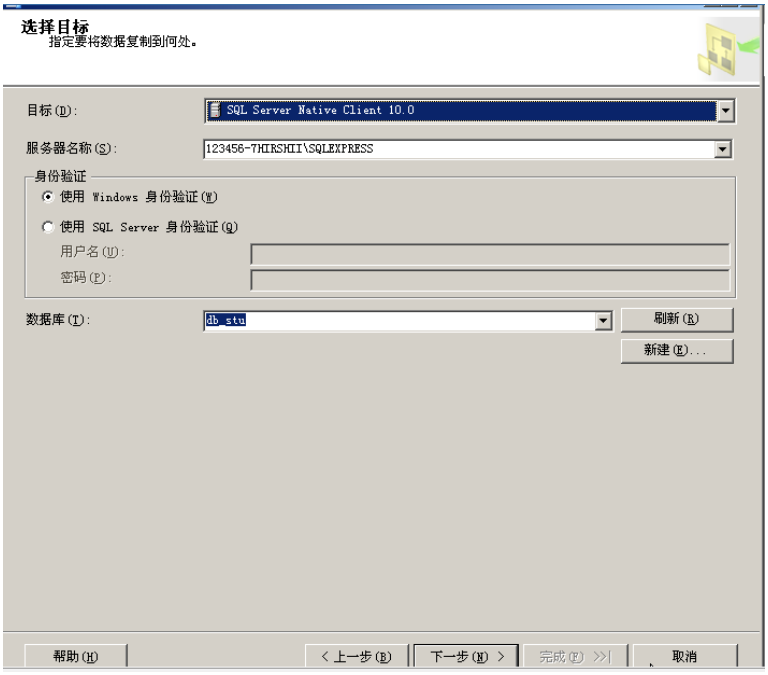


图 8-13　【选择目标】对话框

图 8-14　【指定表复制或查询】对话框

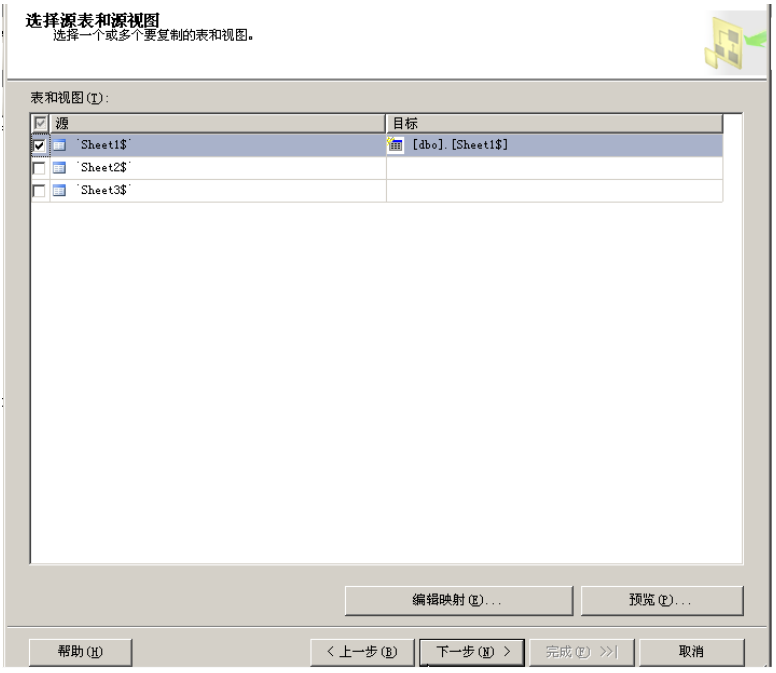


图 8-15　【选择源表和源视图】对话框

图 8-16　【运行包】对话框

图 8-17　【执行成功】对话框

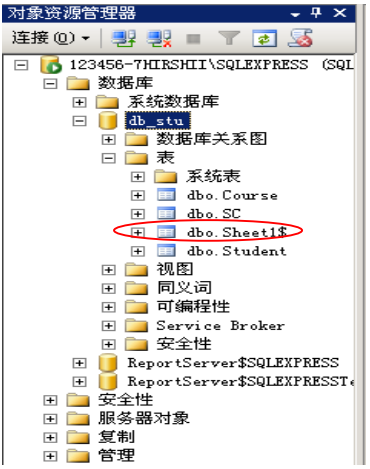


图 8-18　数据表导入成功

从数据库中导出数据和导入数据的步骤相似，这里就不再重复了。

习　　题

1．数据导入/导出的含义是什么？
2．数据库恢复要执行哪些操作？
3．如何使用对象资源管理器和 T-SQL 语句创建和恢复数据库备份？
4．在备份数据库的过程中应避免哪些操作？
5．参照本章中备份数据库的方法，对数据库【db_stu】进行备份。
6．参照本章所学的方法，将一个 Excel 数据表中的数据导入到数据库中。

第 9 章　SQL Server 2008 安全管理

数据库系统的安全性是每个数据库管理员必须认真对待的问题。保护数据不受来自内部和外部的侵害，对于数据库管理员和开发人员来说是一项非常重要的工作。SQL Server 2008 提供了从操作系统、SQL Server 服务器、数据库到对象的多级别安全保护，其中涉及角色、数据库用户、权限等多个与安全性有关的概念。

- 了解 SQL Server 2008 的安全性
- 掌握 SQL Server 2008 的登录模式
- 掌握数据库用户
- 掌握权限的分配

9.1　安全管理概述

就目前而言，大多数数据库管理系统的安全管理机制都还是由运行在某一个特定操作系统平台下的应用程序完成的，SQL Server 也不例外。SQL Server 的安全性机制可以分为以下 4 个等级：

- 操作系统的安全性；
- SQL Server 的登录安全性；
- 数据库的使用安全性；
- 数据库对象的使用安全性。

1．操作系统的安全性

在用户使用客户计算机通过网络实现对 SQL Server 服务器的访问时，用户首先要获得客户计算机操作系统的使用权。

一般来说，在能够实现网络互联的前提下，用户没有必要向运行 SQL Server 服务器的主机进行登录，除非 SQL Server 服务器就运行在本地计算机上。SQL Server 可以直接访问网络端口，所以可以实现对 Windows NT 安全体系以外的服务器及其数据库的访问。

保证操作系统安全性是操作系统管理员或者网络管理员的任务。由于 SQL Server 采用了集成 Windows NT 网络安全性的机制，所以使得操作系统安全性的地位得到了提高，但同时也加大了管理数据库系统安全性和灵活性的难度。

2．SQL Server 的安全性

SQL Server 的服务器级安全性建立在控制服务器登录和密码的基础上。SQL Server 采用了标准的 SQL Server 登录和集成 Windows NT 登录两种方式。无论使用哪种登录方式，用户在登录时提供的登录账号和密码都决定了用户能否去访问 SQL Server。

SQL Server 在服务器和数据库级的安全级别上都设置了角色。SQL Server 允许用户在数据库上建立新的角色，并为该角色授予多个权限，再通过角色将权限赋予 SQL Server 的用户，但是 SQL Server 不允许用户建立服务器级的角色。

3．数据库的安全性

在用户通过 SQL Server 服务器的安全性检查后，将直接面对不同的数据库入口。这是用

户接受的第三次安全性检验。

在建立用户的登录账号信息时，SQL Server 会提示用户选择默认的数据库。以后用户每次连接上服务器后，都会自动转到默认的数据库上。如果用户在设置登录账号时没有指定默认的数据库，则用户的权限将限制在 master 数据库以内。

在默认情况下，数据库的拥有者可以访问该数据库的对象，还可以分配访问权限给别的用户，以便让别的用户也拥有该数据库的访问权限，但是在 SQL Server 中并不是所有的权限都可以自由转让和分配。

需要注意的是，master 数据库存储了大量的系统信息，对系统的安全性和稳定性起着至关重要的作用，所以用户在建立新的登录账号时，最好不要将默认的数据库设置为 master 数据库。而是应该根据用户将要进行的工作，将默认的数据库设置在具有实际操作意义的数据库上。

4．SQL Server 数据库对象的安全性

数据库对象的安全性是核查用户权限的最后一个安全等级。在创建数据库对象的时候，SQL Server 将自动把该数据库对象的拥有权赋予该对象的所有者。对象的拥有者可以实现对该对象的安全控制。

数据对象访问的权限定义了用户对数据库中数据对象的引用、数据操作语句的许可权限。这部分工作通过定义对象和语句的许可权限来实现。

SQL Server 安全模型的 3 个层次对于用户权限的划分不存在包含的关系，但是它们之间并不是孤立的，相邻的层次是通过映射账号建立关联的。例如，用户访问数据的时候经过 3 个阶段的处理。

第一阶段，用户必须登录到 SQL Server 的实例进行身份鉴别，被确认合法才能登录 SQL Server 实例。

第二阶段，用户在每个要访问的数据库里必须要有一个账号，SQL Server 实例将 SQL Server 登录映射到数据库用户账号上，在这个数据库的账号上定义数据库管理和数据对象的访问安全策略。

第三阶段，检查用户是否具有访问数据库对象、执行动作的权限，经过语句许可权限的验证，才能够实现对数据的操作。

9.2　SQL Server 验证模式

为了实现安全性，SQL Server 对用户的访问进行了两个阶段的检验，即验证阶段和许可确认阶段。

1．验证阶段

用户在 SQL Server 上获得对任何数据库的访问权限之前，必须登录到 SQL Server 上且是合法的。SQL Server 或者 Windows 对用户进行验证，如果验证通过，用户就可以连接到 SQL Server 上，否则服务器将拒绝用户登录。

2．许可确认阶段

用户验证通过后，登录到 SQL Server 上，系统检查用户是否有访问服务器上数据的权限。

身份验证阶段用来识别用户的登录账号和验证用户与 SQL Server 相连接的能力。如果验证成功，用户就能连接到 SQL Server 上。SQL Server 能够识别两种登录身份验证机制，即 Windows 身份验证和混合身份验证。每一种身份验证都有一个不同类型的登录账户。

9.2.1　Windows 身份验证模式

当使用 Windows 身份验证连接到 SQL Server 时，Microsoft Windows 将完全负责对客户端进行身份验证。在这种情况下，将按其 Windows 用户账户来识别客户端。当用户通过 Windows 用户账户进行连接时，SQL Server 使用 Windows 操作系统中的信息验证账户名和密码，这是 SQL Server 默认的身份验证模式，比混合模式安全得多。其登录模式如图 9-1 所示。

图 9-1　Windows 身份验证

当使用 Windows 身份验证时，用户账户或 Windows 组会控制用户访问 SQL Server。当进行连接时，用户不用提供 SQL Server 登录账户。SQL Server 系统管理员必须定义用户账户或 Windows 组，并将其作为有效的 SQL Server 登录账户。

在 Windows 身份验证模式下，SQL Server 检测当前使用 Windows 的用户账户，并在系统注册表中查找该用户，以确定该用户账户是否有权限登录。在这种方式下，用户不必提交登录名和密码让 SQL Server 验证。

Windows 身份验证模式有以下几个主要优点：

- 数据库管理员的工作可以集中在管理数据库上面，而不是管理用户账户上。对用户账户的管理可以交给 Windows 去完成；
- Windows 有着更强的用户账户管理工具，可以设置账户锁定、密码期限等。如果不是通过定制来扩展 SQL Server，SQL Server 是不具备这些功能的；
- Windows 的组策略支持多个用户同时被授权访问 SQL Server。

实际上，SQL Server 是从 RPC 协议连接中自动获取登录过程中的 Windows 用户账户信息的。多协议和命名管道自动使用 RPC 协议，因此在客户机和服务器间可以使用 Windows 身份验证模式。

但是，应该注意的是，要使用多协议或者命名管道在客户和服务器之间建立连接，必须满足两个条件中的一个。第一，客户端的用户必须是合法的服务器上的用户账户，服务器能够在自己的域中或者信任域中验证该用户。第二，服务器启动了 guest 账户，但是该方法会带来安全上的隐患，因而不是一个好的方法。如果条件允许，在此我们推荐使用 Windows 身份验证连接到 SQL Server 2008 服务器。

9.2.2　混合身份验证模式

混合验证模式允许以 SQL Server 身份验证模式或者 Windows 身份验证模式来进行验证。使用哪个模式取决于在最初的通信时使用的网络库。如果一个用户使用 TCP/IP Sockets 进行登录验证，则使用 SQL Server 身份验证模式；如果用户使用命名管道，则登录时将使用 Windows 验证模式。这种模式能更好地适应用户的各种环境。如图 9-2 所示为使用 SQL Server 身份验证的界面。

图 9-2　SQL Server 身份验证模式

在使用 SQL Server 身份验证模式时，用户必须提供登录名和密码，SQL Server 通过检查是否注册了该 SQL Server 登录账户，或指定的密码是否与以前记录的密码相匹配来进行身份验证。如果 SQL Server 未设置登录账户，则身份验证将失败，而且用户会收到错误信息。

混合身份验证模式具有如下优点：

- 创建了 Windows NT/2000 之上的另外一个安全层次；
- 支持更大范围的用户，如非 Windows 客户等；
- 一个应用程序可以使用单个的 SQL Server 登录和口令。

9.2.3　设置身份验证模式

在第一次安装 SQL Server 或者使用 SQL Server 连接其他服务器的时候，需要指定验证。对于已指定验证模式的 SQL Server 服务器，在 SQL Server 中还可以进行修改。具体步骤如下。

（1）打开 SQL Server Management Studio，使用 Windows 或 SQL Server 身份验证建立连接。

（2）在【对象资源管理器】窗口中选择服务器名，并在其上单击鼠标右键，在弹出的菜单中选择【属性】命令，打开【服务器属性】窗口，如图 9-3 所示。

（3）选择【安全性】选项，打开【安全性】选项窗口，如图 9-4 所示。在此窗口中可以设置身份验证模式。

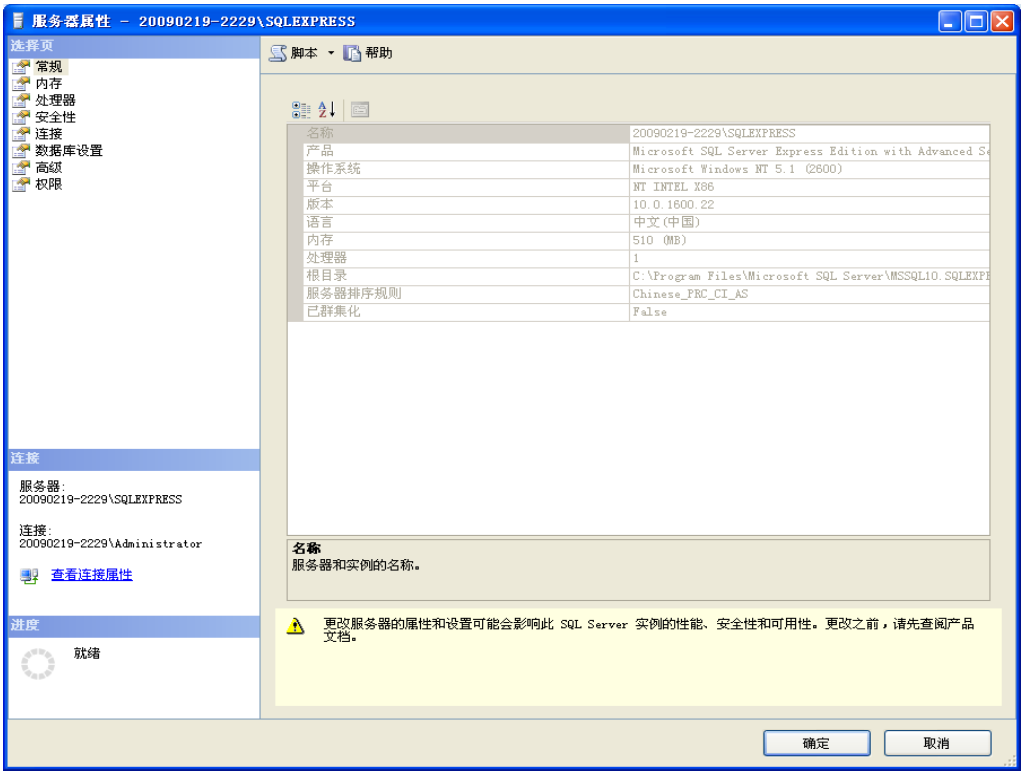


图 9-3　【服务器属性】窗口

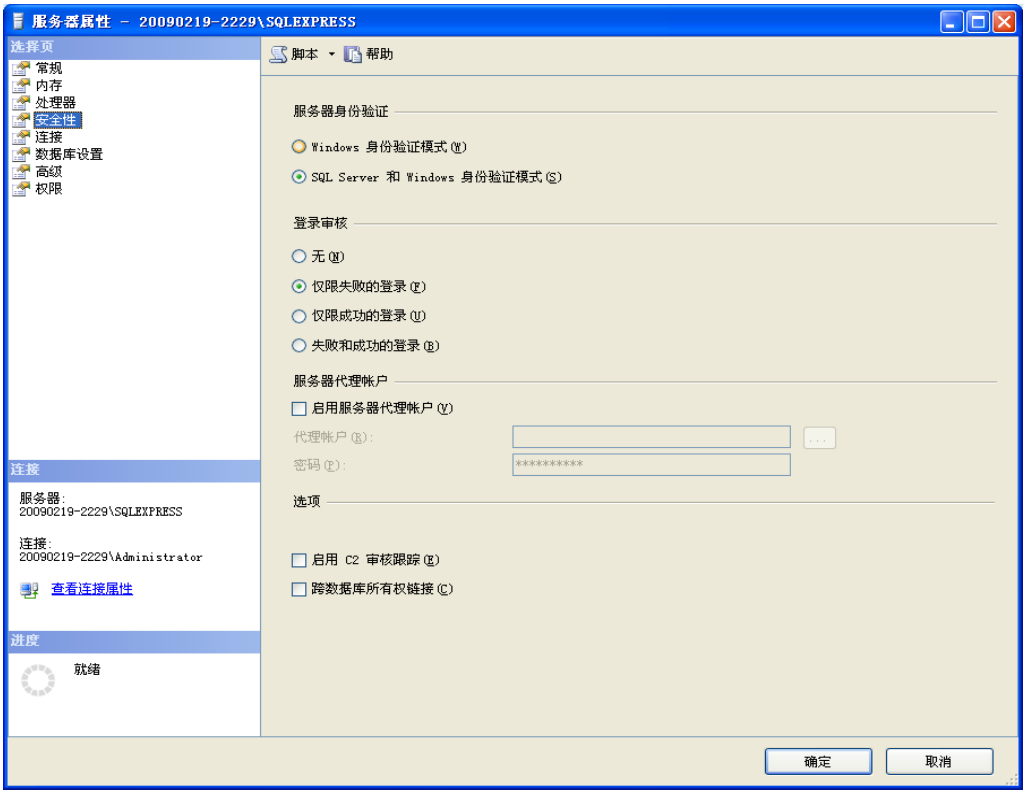


图 9-4　【安全性】选项窗口

9.3　数据库账号

在 SQL Server 中，账号有两种。一种是登记服务器的登录账号，另一种是使用数据库的用户账号。登记账号是让用户登录到 SQL Server 中，登记中本身并不能让用户访问服务器中的数据库。

要访问特定的数据库，还必须具有用户名。用户名在特定的数据库内创建，并关联一个登录名（当一个用户被创建时，必须关联一个登录名）。通过授权给用户来指定用户可以访问的数据库对象的权限。可以这样想象，假设 SQL Server 是一个包含许多房间的大楼，每一个房间代表一个数据库，房间里的资料可以表示数据库对象。则登录名就相当于进入大楼的钥匙，而每个房间的钥匙就是用户名。房间中的资料则根据用户名的不同而有不同的权限。

9.3.1　服务器的登录账号

创建登录账号时需要指出该账号使用的是 Windows 身份验证还是 SQL Server 身份验证。

1．使用 Windows 身份验证的登录

SQL Server 默认的身份验证类型为 Windows 身份验证，如果使用 Windows 身份验证登录 SQL Server，该登录账号必须存在于 Windows NT 或者 Windows 2000 账号数据库中。

打开【SQL Server Management Studio】，展开【安全性】节点，右击【登录名】选项，在菜单上选择【新建登录名】命令，创建 Windows 身份验证登录，如图 9-5 所示。

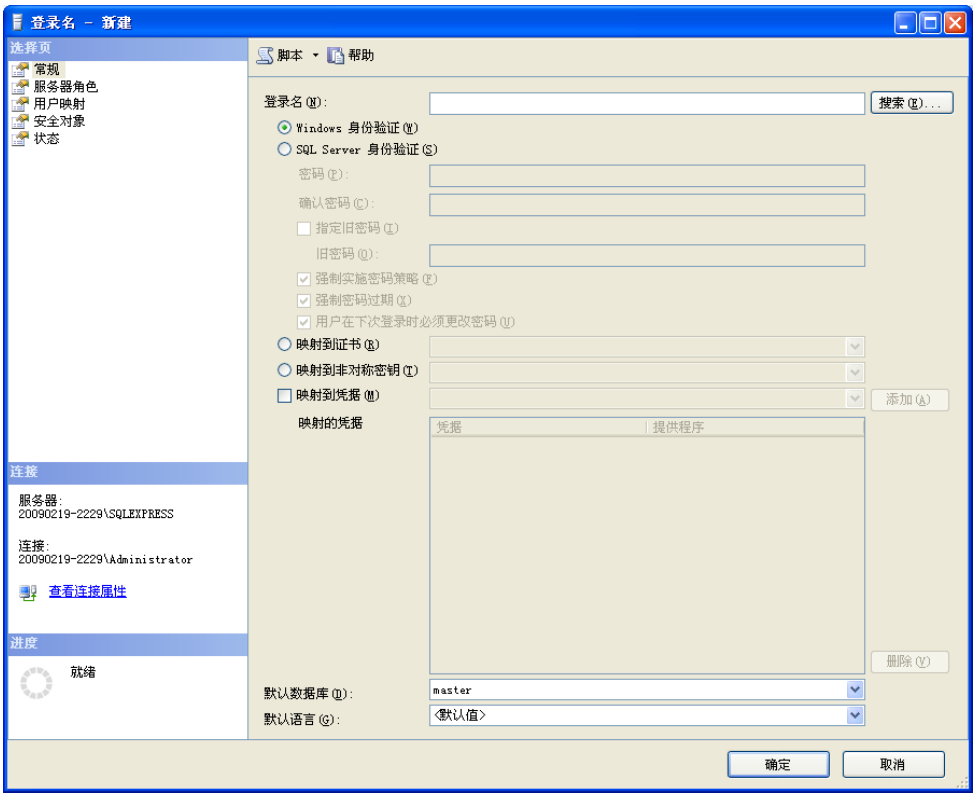


图 9-5　创建 Windows 身份验证

单击【搜索】按钮，弹出【选择用户或组】对话框，如图 9-6 所示。选择 Windows 用户后单击【确定】按钮，将该用户添加为 Windows 身份验证登录。

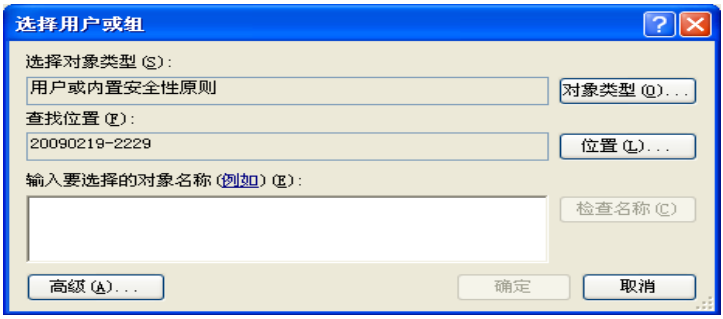


图 9-6　选择 Windows 用户

在创建账号的时候，在【登录名-新建】窗口中可以为每个用户指定一个默认数据库，用户在每次连接到 SQL Server 实例上的时候会默认访问该数据库，如图 9-5 所示。还可以使用 sp_defaultdb 系统存储过程为用户指定默认的数据库。

如果登录名 sa 希望设置默认数据库为 msdb，则执行以下语句：

```
EXECUTE　sp_defaultdb'sa' 'msdb'
```

2．使用 SQL Server 身份验证的登录

如果使用 SQL Server 身份验证，用户必须拥有合法的账号和正确的密码。其中登录密码并不是必须的，但是用户最好设置密码，没有密码的保护，SQL Server 的服务器级安全保护就像没有上锁的大门。

打开【SQL Server Management Studio】，展开【安全性】节点，右击【登录名】选项，在菜单上选择【新建登录名】命令，创建 SQL Server 身份验证登录，如图 9-7 所示。

还可以通过使用 sp-addlogin 系统存储过程创建一个新的 SQL Server 登录，让用户使用 SQL Server 身份验证连接到 SQL Server 实例。使用 sp_droplogin 系统存储过程可以删除由 sp_addlogin 添加的 SQL Server 登录。

下面创建一个 SQL Server 身份验证连接，用户名为 abc，密码为 adminsoft，默认数据库为 db_stu。

```
EXECUTE sp-addlogin'abc'，'adminsoft'，'db_stu'
```

如果要删除 sp-addlogin 添加的 SQL Server 登录 abc，可使用下面的语句：

```
EXECUTE sp_droplogin'abc'
```

提供 SQL Server 身份验证的登录具有支持更大范围用户的特点。应用程序开发人员和数据库用户也许更喜欢 SQL Server 身份验证，因为他们熟悉登录和密码功能。

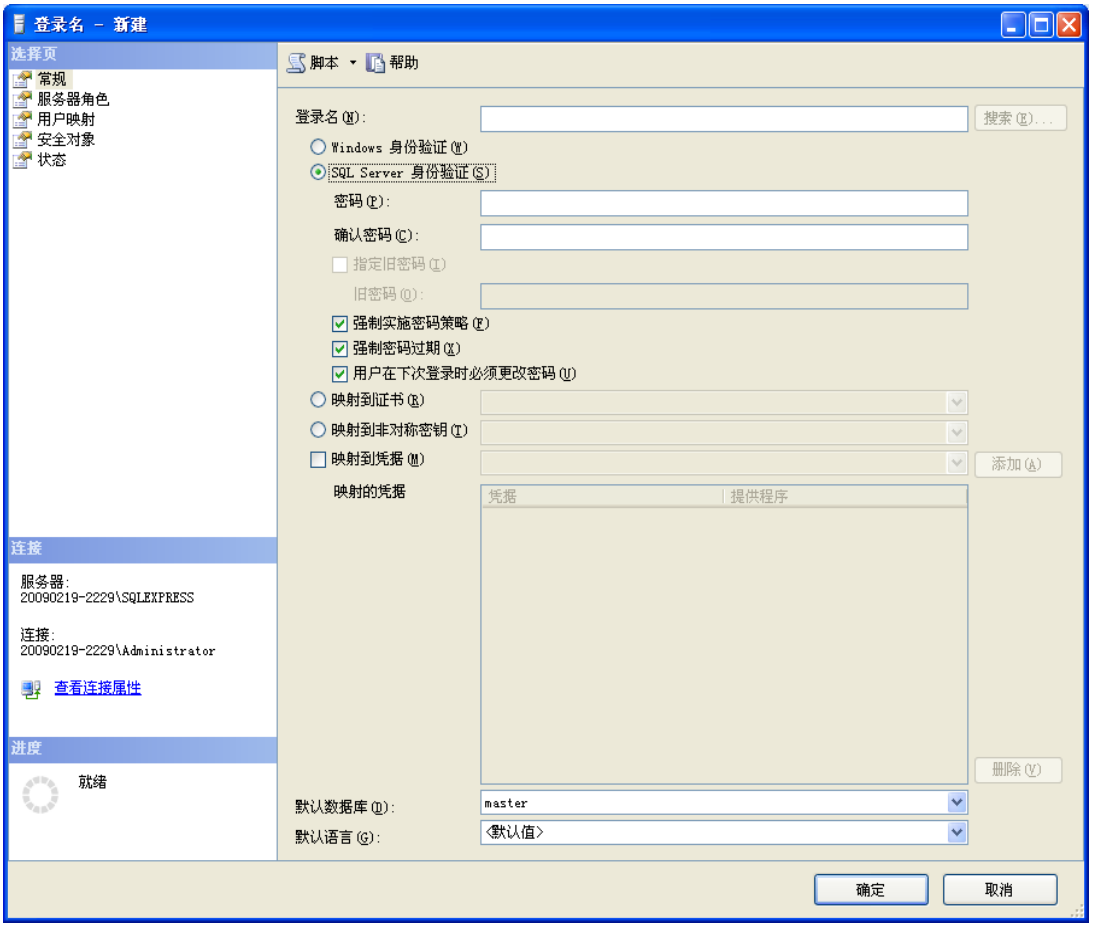


图 9-7　创建 SQL Server 身份验证登录

9.3.2　数据库用户账户

一般情况下，用户登录 SQL Server 实例后，还不具备访问数据库的条件。在用户可以访问数据库之前，管理员必须为该用户在数据库中建立一个数据库账号作为访问该数据库的 ID。这个过程就是将 SQL Server 登录账号映射到需要访问的每个数据库中，这样才能够访问数据库。如果数据库中没有用户账户，则即使用户能够连接到 SQL Server 实例也无法访问该数据库。

1．默认的数据库用户

在 SQL Server 的数据库级别上，存在着两个特殊的数据库用户。这两个数据库用户分别是 dbo 和 guest。

dbo 是数据库的拥有者，它存在于每个数据库下，是数据库的管理员。在安装时就被设置到 model 数据库中，而且不能被删除。dbo 用户对应于创建该数据库的登录账户，所有系统数据库的 dbo 用户都对应于 sa 账户。

注意：当某个 SQL Server 登录账户是 sysadmin 角色的成员并且该登录没有向数据库中映射账号时，系统将会自动将其映射到每个数据库的 dbo 用户中。

guest 用户可以使已经登录到 SQL Server 服务器上的用户能够访问数据库。所有的系统数

据库（model 除外）都有 guest 用户。如果 SQL Server 登录账号没有向要访问的数据库中映射账号，但该数据库下存在 guest 账号，那么用户将以 guest 用户的身份访问数据库，并且拥有管理员对 guest 用户定义的数据库对象的访问权限。如果要添加 guest 用户，则必须使用 sp_grantdbaccess 系统存储过程明确地建立这个用户。

2．添加数据库用户

将登录账户添加为数据库用户后，使用该登录账户登录的 SQL Server 用户就可以实现对数据库的访问了。

添加数据库用户可以用系统存储过程 sp_grantdbaccess 来实现，具体语法如下：

```
sp_grantdbaccess [@loginname =]'login'
[,[@name_in_db=]'name_in_db']
```

其中，参数介绍如下。

- @loginname：映射到新数据库用户的 Windows 组、Windows 登录名或 SQL Server 登录名的名称。
- @name_in_db：新数据库用户的名称 name_in_db 是 OUTPUT 变量，其数据类型为 sysname，默认值为 NULL。如果不指定，则使用 login。如果指定值为 NULL 的 OUTPUT 变量，则@name_in_db 将设置为 1ogin。Name_in_db 不能存在于当前数据库中。

下面的例子建立了一个 SQL Server 的登录账户，然后将该账户添加为【db_stu】数据库的用户。

```
EXEC sp_addlogin admin，admin123，db_stu
GO
USE db_stu
GO
EXEC sp_grantdbaccess admin
GO
```

3．删除数据库用户

可以使用系统存储过程 sp_revokedbaccess 来删除数据库用户，更确切地说是断开 SQL Server 的登录账户与数据库用户之间的对应关系，其基本语法如下：

```
Sp_revokedbaccess[@name_in_db=]'name'
```

下面这个例子，断开了账户 admin 与【db_stu】之间的对应关系：

```
USE db_stu
GO
EXEC sp_revokedbaccess admin
GO
```

9.4　固定服务器角色

角色是 SQL Server 引进的用来集中管理数据库或服务器权限的概念。数据库管理员将操

作数据库的权限赋予角色，就好像把职权赋予了一个官位。然后，数据库管理员可以将角色再赋给数据库用户或登录账户，从而使数据库用户或登录账户拥有了相应的权限。

SQL Server 在服务器级提供了固定服务器角色，在数据库级提供了数据库级角色。用户可以修改数据库级的权限，也可以创建新的数据库角色，再分配权限给新的角色。下面介绍服务器角色，数据库角色将在后面介绍。

9.4.1　服务器角色概述

固定服务器角色是 SQL Server 在安装时就创建好的用于分配服务器级管理权限的实体。使用系统存储过程 sp_helpsrvrole 可以浏览固定服务器角色的内容。

```
EXEC sp_helpsrvrole
GO
```

还可以通过 SQL Server Management Studio 来浏览固定服务器角色。打开【SQL Server Management Studio】，在【对象资源管理器】窗口中选择【安全性】选项并展开其节点，在其节点下展开【服务器角色】节点，如图 9-8 所示。

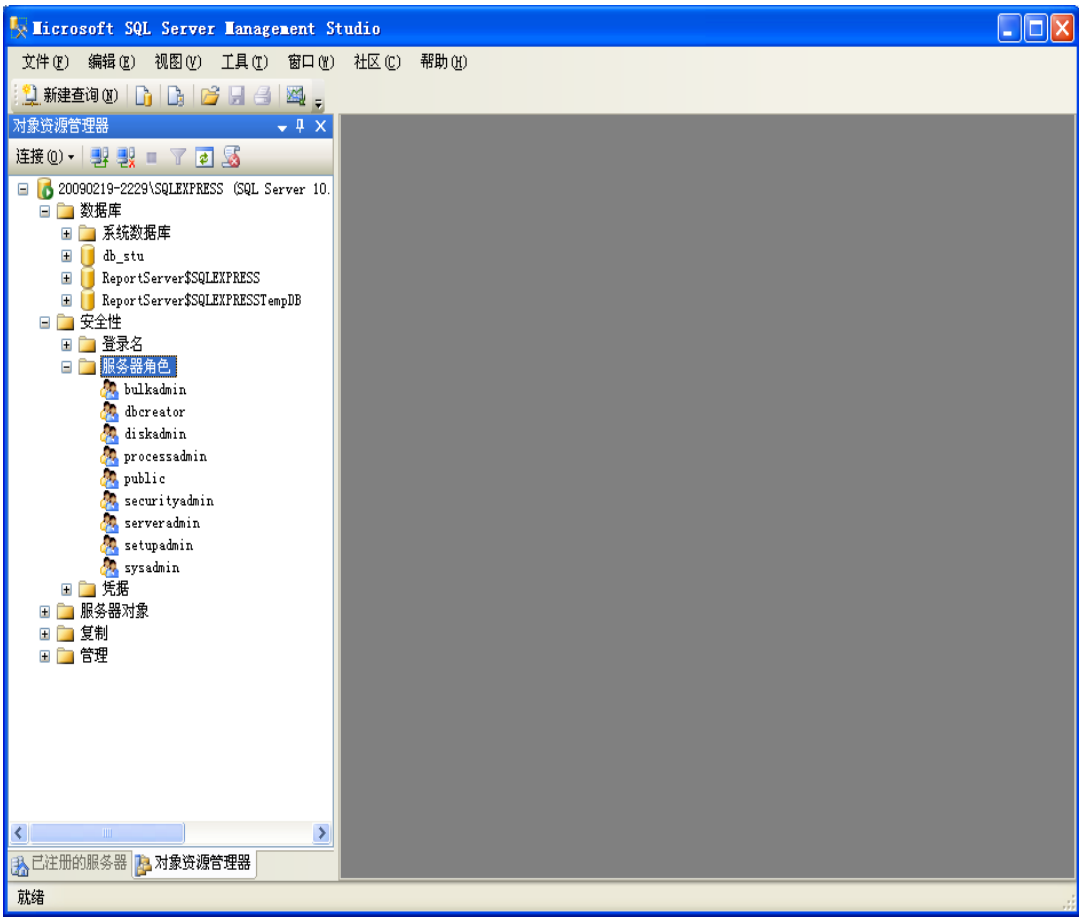


图 9-8　固定服务器角色

当几个用户共同完成一个公共的活动时，管理员就可以将它们集中到一个称为“角色”的

单元中，并且给指定的角色分配一次权限即可，用户不能增加、修改和删除固定的服务器角色。下面从高到低列出了固定的服务器角色的管理性访问权。

- sysadmin：这个服务器角色的成员可在 SQL Server 中执行任何任务。这个角色仅适合数据库管理员。
- serveradmin：这些用户可以设置服务器级的配置选项，如 SQL Server 可以使用多大内存通过网络发送多少信息。它们还可以关闭服务器。
- setupadmin：这个角色的成员可以安装、复制和管理扩展存储过程。
- securityadmin：这些用户可管理安全性问题，如创建与删除登录、阅读审核日志，以及给用户授予创建数据库的权限。
- processadmin：SQL Server 能够多任务化，也就是说，它可以通过执行多个进程做多件事情。例如，SQL Server 可以生成一个进程用于向高速缓存写数据，同时生成另一个进程用于从高速缓存中读取数据。这个角色的成员可以结束进程。
- dbcreator：这些用户可以创建和修改数据库。
- diskadmin：这些用户可管理磁盘文件，如镜像数据库和添加备份设备。
- bulkadmin：这个角色的成员可以执行 BULK IN SERT 语句，这条语句允许它们从文本文件中将数据导入到 SQL Server 数据库中。

sysadmin 角色的成员是 SQL Server 的系统管理员，该角色下的成员能够在 SQL Server 实例中执行全部动作。

9.4.2 服务器角色管理

固定服务器角色的信息存储在 master 数据库的 sysxlogins 表中，当用户将某个登录账户增加到服务器角色中的时候，sysxlogins 表中该登录账户的相应行将进行更新，以指示该登录账户属于服务器角色的成员，同时具有该角色的许可权限。

使用 Microsofi SQL Server 2008 的管理器可以实现固定服务器角色的管理。步骤如下。

（1）打开【SQL Server Management Studio】，在【对象资源管理器】窗口中选择【安全性】选项。

（2）展开【安全性】节点，选择该节点下的【服务器角色】选项，可以看到所有的固定服务器角色，如图 9-8 所示。

（3）选中要指定的固定服务器角色，单击鼠标右键，从弹出的菜单中选择【属性】命令，则系统弹出如图 9-9 所示的窗口，显示所有分配了该固定服务器角色的登录账户。

（4）可以单击图 9-9 中的【添加】按钮进行选择，以便添加更多的登录账户，也可以选择一个登录账户再单击【删除】按钮，断开该登录账户与固定服务器角色的映射关系。

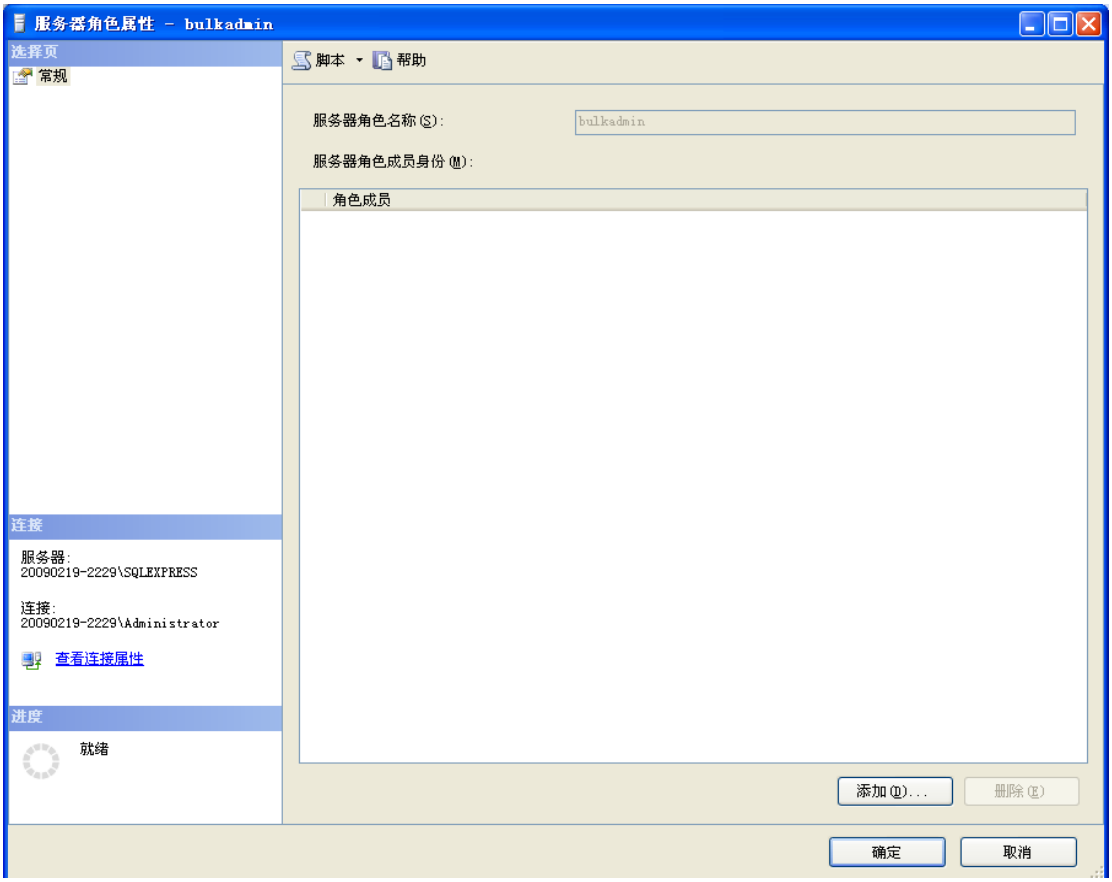


图 9-9　【服务器角色属性】窗口

9.5　数据库角色

管理员可以将用户添加到角色，对于用户数据众多或安全系统复杂的数据库，角色可以简化其安全管理。数据库角色分为固定数据库角色、用户定义角色和应用程序角色。固定数据库角色预定义了数据库的安全管理权限和对数据对象的访问权限。用户定义角色由管理员创建并且定义了对数据对象的访问权限。应用程序角色规定了某个应用程序的安全性，用来控制通过某个应用程序对数据的间接访问。当在 SQL Server 中添加新用户账户或者更改现有用户的权限时，可以向 SQL Server 数据库角色添加此用户，而不要直接将权限应用到账户上。用户账户可以是同一数据库中任意多个角色的成员，并且可以拥有每个角色的适当权限。

9.5.1　固定数据库角色

固定数据库角色存在于每个数据库中，在数据库级别上提供管理特权分组。管理员可将任何有效的数据库用户添加为固定数据库角色成员。每个成员都获得应用于固定数据库角色的权限。用户不能增加、修改和删除固定数据库角色。

SQL Server 在数据库级放置了固定数据库角色来提供最基本的数据库权限的综合管理。在数据库创建时，系统默认创建了 10 个固定数据库角色，我们可以通过 SQL Server 2008 中的管

理器工具来浏览固定数据库角色。具体操作步骤如下。

（1）打开【SQL Server Management Studio】，在【对象资源管理器】窗口中选择任意一个数据库名。

（2）展开该数据库的节点，选择【安全性】选项并展开其节点。

（3）在【安全性】节点下面选择【角色】选项展开其节点，然后选择【数据库角色】选项展开，此时就可以看到系统默认的固定数据库角色了，如图 9-10 所示。

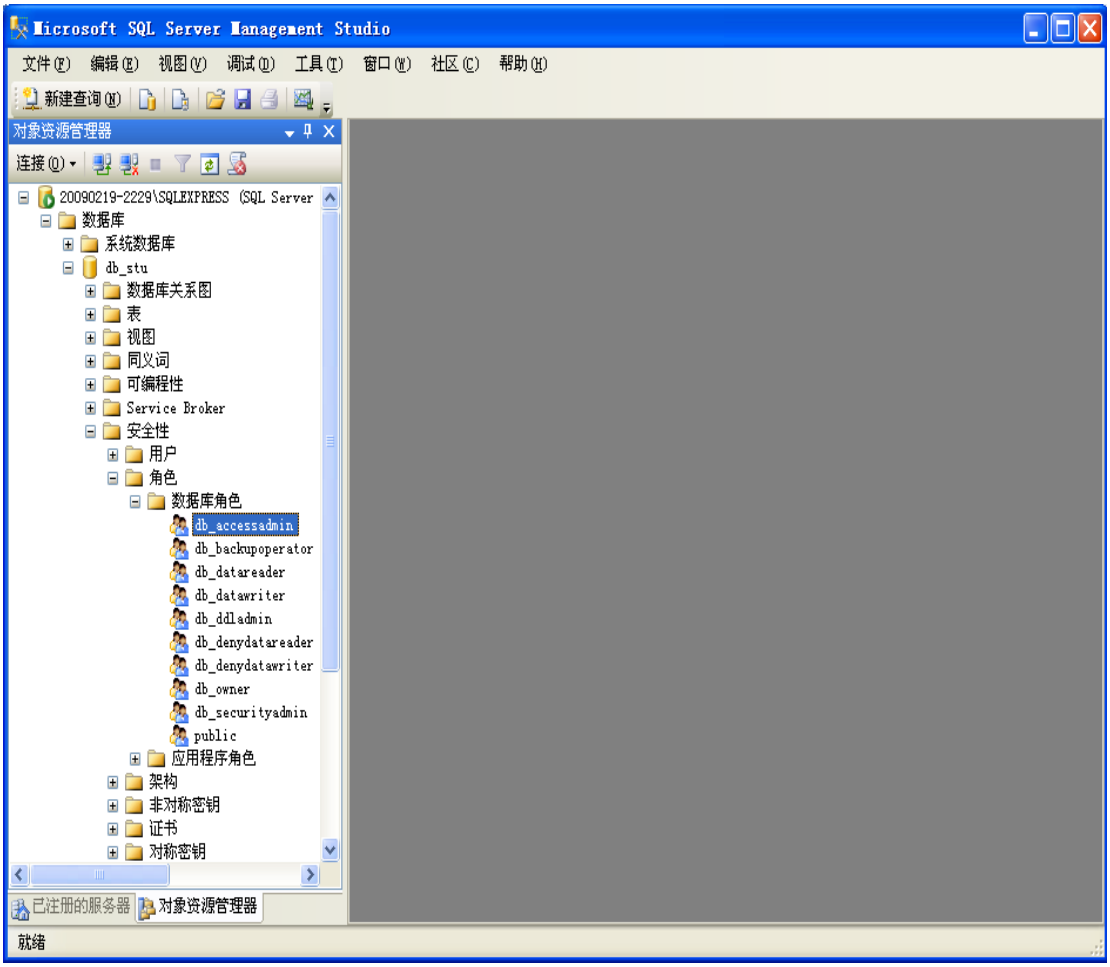


图 9-10　在管理器工具中浏览固定数据库角色

固定数据库角色提供了数据库级别的管理权限分组功能。下面描述 SQL Server 中的固定数据库角色。

- public：为数据库用户维护所有的默认许可权限。每个数据库用户都属于 public 角色的成员。
- db_owner：进行所有数据库角色的活动，以及数据库中的其他维护和配置活动。该角色的权限跨越所有其他的固定数据库角色。
- db_accessadmin：在数据库中添加或删除 Windows 用户和组，以及 SQL Server 用户。
- db_datareader：查看来自数据库中所有用户表的全部数据。
- db_datawriter：添加、更改或删除来自数据库中所有用户表的数据。

- db_ddladmin：添加、修改或删除数据库中的对象（运行所有 DDL）。
- db_securityadmin：管理 SQL Server 数据库角色的角色和成员，并管理数据库中的语句和对象权限。
- db_backupoperator：有备份数据库的权限。
- db_denydatareader：拒绝选择数据库数据的权限。
- db_denydatawriter：拒绝更改数据库数据的权限。

我们还可以通过 SQL Server 2008 的管理工具来完成对数据库角色的管理，具体步骤如下。

（1）打开【SQL Server Management Studio】，在【对象资源管理器】窗口中选择任意一个数据库名。

（2）展开该数据库节点下的【安全性】节点，选择【服务器】节点下的【数据库角色】选项。

（3）选中指定的数据库角色，单击鼠标右键，从弹出的菜单中选择【属性】命令，则系统弹出如图 9-11 所示的窗口，显示所有分配了该数据库角色的登录账户。

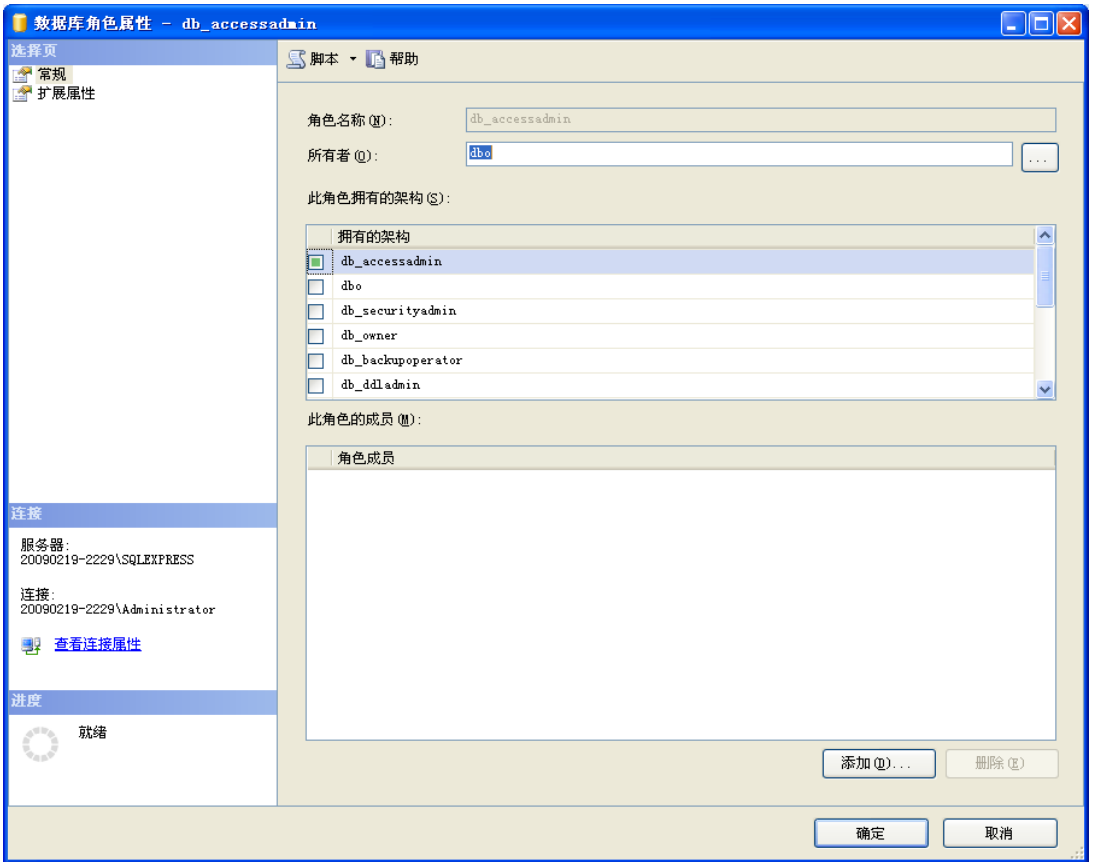


图 9-11　【数据库角色属性】窗口

（4）单击【添加】按钮，可从弹出的对话框中选取登录账户，添加到该数据库角色中去。

public 数据库角色是 SQL Server 的基本操作，所有的数据库用户都属于 public 角色。public 角色的特点如下：

- 替数据库用户捕获所有的数据库默认权限；

- 不能将 public 角色分配给任何用户、工作组，因为所有的用户都默认为属于该角色；
- public 角色存在于每一个数据库中，包括系统数据库和用户建立的数据库；
- 不允许被删除。

在所有数据库用户中，特殊用户 dbo 具有最高的管理权限，它被认为是所有数据库对象的拥有者，可以访问所有的数据库对象，可以在数据库范围内执行一切操作。在固定服务器角色和 dbo 之间有着一种奇特的对应关系。任何被赋予 syadmin 固定服务器角色的用户都映射着每个数据库的特殊用户 dbo。所有 sysadmin 成员创建的数据库对象都自动拥有 dbo 用户。

提示：只有由 sysadmin 固定服务器成员创建的数据库对象才属于 dbo。由任何其他用户（包括 db_owner 固定数据库角色）创建的对象只属于创建对象的用户而不属于 dbo。如果数据库设有 guest 数据库用户，则任何连接上 SQL Server 服务器，但不是本地数据库成员或者不是 sysadmin 固定服务器角色成员的用户所创建的数据库对象都属于用户 guest。

9.5.2　自定义数据库角色

通过创建用户自定义数据库角色，可以建立具有某种公共许可权限的用户组。

我们可以用 SQL Server 2008 的管理工具完成对数据库角色的创建、删除和授权。步骤如下。

（1）展开指定的数据库节点。

（2）选择指定的数据库下面的【安全性】选项，展开【安全性】节点，选择【角色】选项。

（3）展开【角色】选项下的节点，选择【数据库角色】选项，单击鼠标右键，从弹出的菜单中选择【新建数据库角色】命令，系统弹出如图 9-12 所示的窗口。

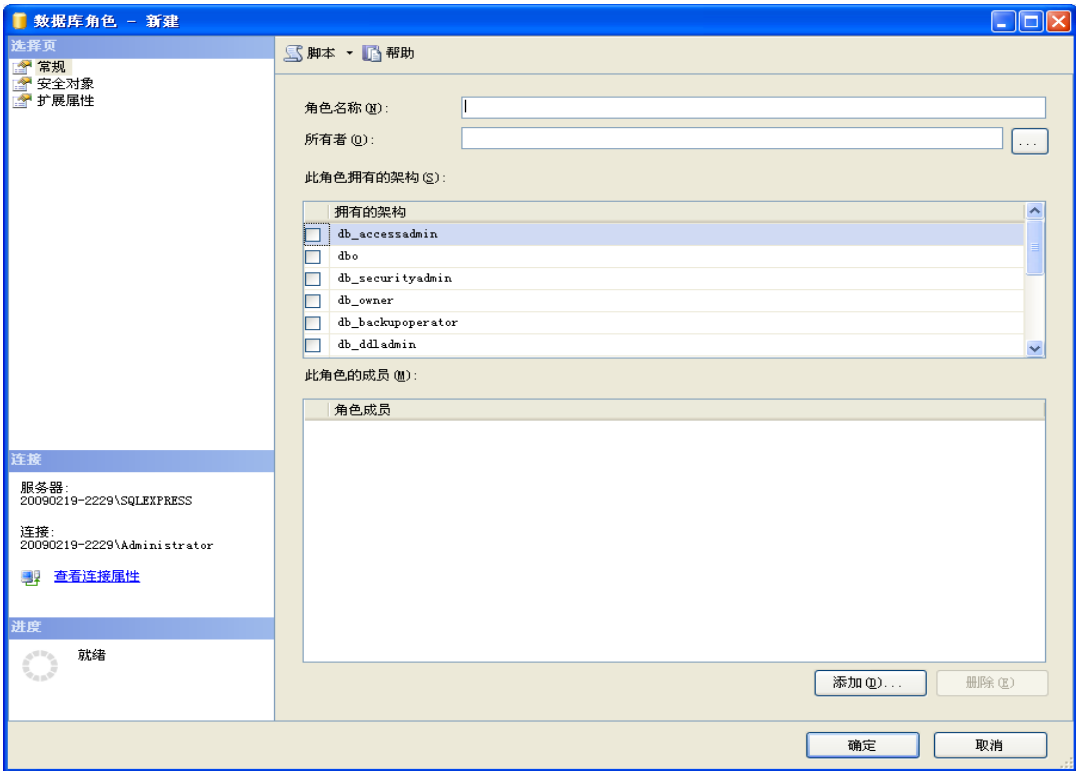


图 9-12　新建数据库角色

（4）用户可以在窗口中输入数据库角色的名字，也可以单击【添加】按钮添加数据库成员。

（5）单击【确定】按钮完成数据库角色的创建。

9.5.3　应用程序角色

应用程序角色是用户定义数据库角色的一种形式。与固定数据库角色不同，它规定了某个应用程序的安全性，用来控制通过某个应用程序对数据的间接访问。例如，管理员允许雇员使用雇员处理程序输入新员工、离职员工和打印统计报表等。

此外，有时可能希望限制用户只能通过特定应用程序（如使用 SQL 查询分析器）来访问数据或防止用户直接访问数据。限制用户的这种访问方式将禁止用户使用应用程序执行编写质量差的查询，以免对整个服务器的性能造成负面影响。

创建应用程序角色的过程与创建数据库角色的过程一样，如图 9-13 所示为应用程序角色的创建窗口。

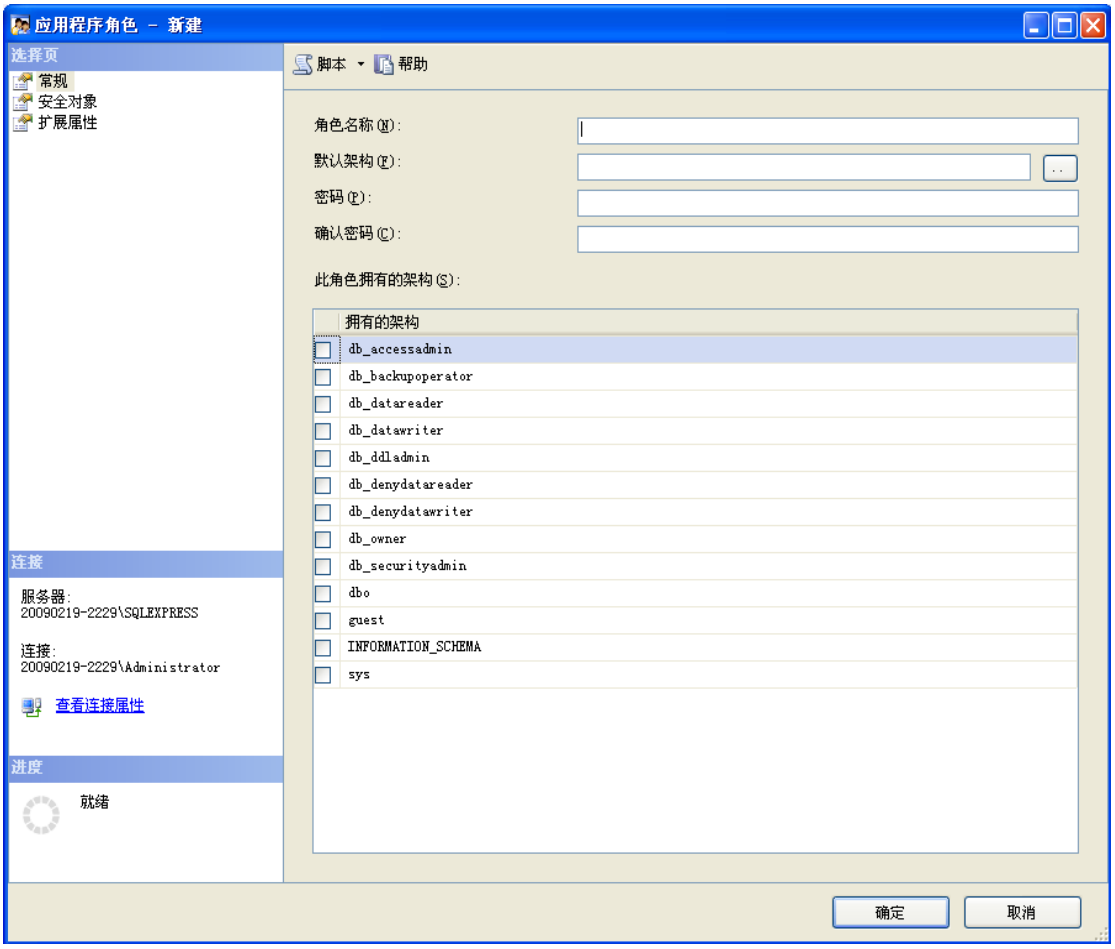


图 9-13　创建应用程序角色

应用程序角色和固定数据库角色的区别有如下 4 点：

- 应用程序角色不包含任何成员。不能将 Windows 组、用户和角色添加到应用程序角色中。

- 当应用程序角色被激活以后，这次服务器连接将暂时失去所有应用于登录账户、数据库用户等的权限，而只拥有与应用程序相关的权限。在断开本次连接以后，应用程序失去作用。
- 默认情况下，应用程序角色是非活动的，需要密码激活。
- 应用程序角色不使用标准权限。

应用程序角色是一个数据库主体，它使应用程序能够用其自身的、类似用户的特权来运行。使用应用程序角色，可以只允许通过特定应用程序连接的用访问特定数据。与数据库角色不同的是，应用程序角色默认情况下不包含任何成员，而且是非活动的。应用程序角色使用两种身份验证模式，可以使用 sp_setapprole 来激活，并需要 guest 密码。因为应用程序角色是数据库级别的主体，所以它们只能通过其他数据库授予的 guest 用户账户的权限来访问这些数据库。因此，任何已禁用 guest 用户账户的数据库对其他数据库中的应用程序角色都是不可访问的。

9.6　数据库权限

权限提供了一种方法来对特权进行分组，并控制实例、数据库和数据库对象的维护和实用程序的操作。用户可以具有授予一组数据库对象的全部特权的管理权限，也可以具有授予管理系统的全部特权但不允许存取数据的系统权限。

9.6.1　权限概述

当数据库对象刚刚创建完后，只有拥有者可以访问该数据库对象。任何其他用户想访问该对象必须首先获得拥有者赋予他们的权限。拥有者可以授予权限给指定的数据库用户，这种权限被称为对象权限。

对于表和视图，拥有者可以授予数据库用户 INSERT、UPDATE、DELETE、SELECT 和 REFERENCES 5 种权限。

在数据库用户要对表执行相应的操作之前，必须事先获得相应的操作权限。例如，如果用户想浏览表中的数据，则它必须首先获得拥有者授予的 SELECT 权限。

提示：REFERENCES 权限允许别的表的拥有者引用本表的列作为外键约束。

存储过程的拥有者可以授予 EXECUTE 权限给别的数据库用户。如果基本表的拥有者不希望其他的用户直接访问基本表的数据，就可以在基本表上建立视图或创建访问基本表的存储过程，然后授予使用视图或存储过程的权限给用户。这样就可以让用户在不直接访问基本表的基础上实现对基本表数据的有限访问。这是 SQL Server 不允许用户访问未授权数据的基本机制之一。

数据库拥有者还可以授予执行某些 Transact-SQL 命令的权限。这种权限在 SQL Server 中被称为命令权限。例如，CREATE TABLE 和 CREATE VIEW。这些命令只有特定的用户（如 dbo）可以使用。如果 dbo 希望别的用户也可以创建表格和视图，必须首先授予执行这些命令的权限给那些用户。

注意：具有 sysadmin 固定服务器角色的用户和数据库对象的拥有者，默认拥有数据库对象所有者的操作权限。

CREATE DATABASE 权限一般不授予用户，它只为服务器管理员保留。

9.6.2　管理权限

对于用户或角色权限的操作有以下 3 种状态：授予、撤销和拒绝。用户和角色的权限以记录的形式存储在各个数据库的 sysprotects 系统表中。

1．授予权限

为了允许用户执行某些活动或者操作数据，需要授予他们相应的权限，可使用 GRANT 语句进行授权活动。授予权限的基本语法如下：

```
GRANT
{ALL[PRIVILEGES]|permission[，．．．n]
[(column[,．．n])]ON{table|view}| ON{table|view}[(column[，．．．n])]
| ON{Stored_procedure}
}
TO security_account[，．．．n]
```

授予命令权限的基本语法如下：

```
GRANT
  {ALL|statement[，…n]}
  TO security_account[，．．．n]
```

其中，对各个参数的介绍如下。

- ALL：表示授予所有可以应用的权限。其中在授予命令权限时，只有固定的服务器角色 sysadmin 成员可以使用 ALL 关键字；而在授予对象权限时，固定服务器角色成员 sysadmin、固定数据库角色 db_owner 成员和数据库对象拥有者都可以使用关键字 ALL。
- statement：表示可以授予权限的命令。例如，CREATE DATABASE。
- permission：表示在对象上执行某些操作的权限。
- column：在表或视图上允许用户将权限局限到某些列上，column 表示列的名字。
- security_account：定义被授予权限的用户单位。security_accunt 可以是 SQL Server 数据库用户，可以是 SQL Server 的角色，还可以是 Windows 的用户或工作组。

下面的例子中使用语句授予 public 角色对数据库【student】表的 INSERT、UPDATE 和 SELETE 权限：

```
USE db_stu
GO
GRANT INSERT，UPDATE，DELETE
ON student
TO public
GO
```

从以上语句可以看出【student】表被授予了 INSERT、UPDATE 和 DELETE 权限；也可以看出【student】表中执行 INSERT、UPDATE 和 DELETE 等操作的权限授予了 public 角色。

提示：权限只能授予本次数据库的用户，如果将权限授予了 public 角色，则数据库里的所有用户都将默认获得该项权限。

2．撤销权限

通过撤销某种权限可以停止以前授予或拒绝的权限。使用 REVOKE 语句可撤销以前授予或拒绝的权限。使用撤销类似于拒绝，但是撤销权限是删除已授予的权限，并不妨碍用户、组或角色从更高级别继承已授予的权限。

撤销对象权限的基本语法如下：

```
REVOKE[GRANT OPTION FOR]
{ALL[PRIVILEGES]l|permission[，．．．n]}
{
[(column[，．．．n])]ON{table|view)|ON{table|view}
[(column[，．．．n])]
|{stored procedure}
}
{TO|FROM}
security account[，．．．n]
[CASCADE]
```

撤销命令权限的语法是：

```
REVOKE　{ALL|Statement[，．．．n]}
FROM security_account[，．．．n]
```

撤销权限的语法基本上与授予权限的语法相同，其中 CASCADE 应用在授予权限时使用了 WITH GRANT OPTION 的情况。如果该用户将被授予的权限授予了其他的用户，则使用 CASCADE 关键字，将撤销所有这些已经授予的权限。

下面的例子中将使用语句撤销 public 角色对“学生信息管理系统”所拥有的权限。

```
USE db_stu
GO
REVOKE SELECT，UPDATE，DELETE
FROM student
GO
```

3．拒绝权限

在授予了用户对象权限以后，数据库管理员可以根据实际情况在不撤销用户访问权限的情况下，拒绝用户访问数据库对象。

拒绝对象权限的基本语法是：

```
DENY
{ALL[PRIVILEGES]|permission[，．．．n]}
{[(column[，．．．n])]ON{table|vi ew}
|ON{table|view)[(column[，．．．n])]
|ON {stored_procedure|extended_procedure}
}
TO security_acount[，．．．n]
```

```
[CASCADE]
```

拒绝命令权限的基本语法是：

```
DENY{ALL|statement[，．．．n]}
TO security_account[，．．．n]
```

拒绝访问的语法要素与授予权限和撤销权限的语法要素意义完全一致。下面的例子是 DENY 命令最常用的情况：

```
USE db_stu
GO
GRANT　INSERT
ON student
TO public
GO
DENY　INSERT
ON student
TO guest
```

这个例子首先将在【db_stu】的【student】表中执行 INSERT 操作的权限授予了 public 角色，这样所有的数据库用户都拥有了该项权限。然后，又拒绝了用户 guest 拥有该项权限。

注意：如果使用了 DENY 命令拒绝某用户获得某项权限，那么即使该用户后来又加入了具有该项权限的某工作组或角色，该用户也将依然无法使用该项权限。

9.6.3　继承权限

用户既可以执行单独被授予的权限，也可以继承和执行对它们所属的角色授权的所有动作。例如，用户 login 是数据库角色 test 的成员，【student】表对于数据库用户 login 撤销了 SELECT 权限，对于数据库角色 test 授予了 SELECT 权限，那么 login 对于【student】表的 SELECT 操作最终还是授予权限的。

DENY 语句能够阻止用户执行相应的动作。无论权限是否直接授予用户，DENY 语句都能够最终决定用户所具有的权限。例如，用户 newlogin 是数据库角色 test 的成员，【student】表对于用户 newlogin 授予了 UPDATE 权限，对于数据库角色 test 拒绝了 UPDATE 权限。这时，newlogin 除了拥有【student】表授予的 UPDATE 权限以外，还继承了【student】表对 test 角色的拒绝 UPDATE 权限。最终，newlogin 对于【student】表的 UPDATE 操作是拒绝权限的。

习　　题

1．描述 Windows 身份验证和混合身份验证模式。
2．简述固定服务器角色的作用。
3．权限的状态有哪几种，区别是什么？
4．SQL Server 2008 有哪几种角色及作用？
5．如何创建一个数据库账户并指定角色？

第 10 章　其　　他

这一章主要讨论 SQL Server 其他的一些概念，包括复制、事务和 SQL Server 2008 中的系统自动化的基础知识。

- 掌握复制的概念
- 掌握事务的概念和各语句的含义及其用法
- 掌握 SQL Server 2008 中的系统自动化的基础知识，包括如何配置“数据库邮件”，如何配置操作员、警报和作业，以及如何使用“维护计划向导”。

10.1　复　　制

复制是一组技术，它是指将数据和数据库对象从一个数据库复制和分发到另一个数据库，然后在数据库间进行同步，以维持一致性的过程。使用复制可以在局域网和广域网、拨号连接、无线连接和 Internet 上将数据分发到不同位置，以及分发给远程或移动用户。

在 Microsoft SQL Server 2008 系统中提供了下列可在分布式应用程序中使用的复制类型。

1．事务性复制

事务性复制通常从发布数据库对象和数据的快照开始。创建了初始快照后，在发布服务器上所做的数据更改和架构修改通常在修改发生时（几乎实时）便传递给订阅服务器。数据更改将按照其在发布服务器上发生的顺序和事务边界应用于订阅服务器，因此，在发布内部可以保证事务的一致性。

事务性复制通常用于服务器到服务器环境中，在以下几种情况下适合采用事务性复制。

- 希望发生增量变更时将其传播到订阅服务器。
- 从发布服务器上发生更改到更改内容到达订阅服务器，应用程序需要这两者之间的滞后时间较短。
- 应用程序需要访问中间数据状态。例如，如果某一行更改了 6 次，事务性复制将允许应用程序响应每次更改（如激发触发器），而不是只响应该行最终的数据更改。
- 发布服务器有大量的插入、更新和删除活动。
- 发布服务器或订阅服务器不是 SQL Server 数据库（如 Oracle）。

默认情况下，事务性发布的订阅服务器应视为只读，因为更改将不会传播回发布服务器。但是事务性复制确实提供了允许在订阅服务器上进行更新的选项。

事务性复制由 SQL Server 快照代理、日志读取器代理和分布代理实现。快照代理准备快照文件（其中包含了已发布表和数据库对象的架构和数据），然后将这些文件存储在快照文件夹中，并在分布服务器中的分布数据库中记录同步作业。

2．合并复制

与事务性复制相同，合并复制通常也是从发布数据库对象和数据的快照开始，并且用触发器跟踪在发布服务器和订阅服务器上所做的后续数据更改和架构修改。订阅服务器在连接到网络时将与发布服务器进行同步，并交换自上次同步以来发布服务器和订阅服务器之间发生更改的所有行。

合并复制通常用于服务器到客户端的环境中。合并复制适用于下列几种情况。

- 多个订阅服务器可能会在不同时间更新同一数据，并将其更改传到发布服务器和其他订阅服务器。
- 订阅服务器需要接收数据，脱机更改数据，并在以后再与发布服务器和其他订阅服务器同步更改。
- 每个订阅服务器都需要不同的数据分区。
- 可能会发生冲突，并且在冲突发生时，需要具有检测和解决冲突的能力。
- 应用程序需要最终的数据更改结果，而不是访问中间数据状态。例如，如果在订阅服务器与发布服务器进行同步之前，订阅服务器上的行更改了 5 次，则该行在发布服务器上仅更改一次来反映最终数据更改（也就是第五次更改的值）。
- 合并复制允许不同站点自主工作，并将更新合并成一个统一的结果。由于更新是在多个节点上进行的，同一数据可能由发布服务器和多个订阅服务器进行了更新。因此，在合并更新时可能会产生冲突，合并复制提供了多种处理冲突的方法。

合并复制由 SQL Server 快照代理和合并代理实现。如果发布未经筛选或使用静态筛选器，快照代理将建立单个快照。如果发布使用参数化筛选器，则快照代理将为每个数据分区创建一个快照。合并代理将初始快照用于订阅服务器。它还将合并自初始快照创建后发布服务器或订阅服务器上所发生的增量数据更改，并根据所配置的规则检测和解决任何冲突。

3．快照复制

快照复制将数据以特定时刻的瞬时状态分布，而不监视对数据的更新。发生同步时，将生成完整的快照并将其发送到订阅服务器。

当符合以下一个或多个条件时，使用快照复制是最合适的：

- 很少更改数据；
- 在一段时间内允许具有相对发布服务器已过时的数据副本；
- 复制少量数据；
- 在短期内出现大量更改。

在数据更改量很大但很少发生时，快照复制是最合适的。例如，如果某销售组织维护一个产品价格列表，这些价格每年要在固定时间进行一两次完全更新，那么建议在数据更改后复制完整的数据快照。对于给定的某些类型的数据，更频繁的快照可能也比较适合。例如，如果一天中在发布服务器上更新相对小的表，但可以接受一定的滞后时间，则可以在夜间以快照形式传递更改。

发布服务器上快照复制的连续开销低于事务性复制的开销，因为不用跟踪增量更改。但是，如果要复制的数据集非常大，那么若要生成和应用快照，将需要使用大量资源。评估是否使用快照复制时，需要考虑整个数据集的大小及数据的更改频率。

默认情况下，以上 3 种复制都使用快照初始化订阅服务器。SQL Server 快照代理始终生成快照文件，但传递文件的代理因使用的复制类型而异。快照复制和事务性复制使用分发代理传递文件，而合并复制使用 SQL Server 合并代理传递文件。快照代理在分发服务器上运行。对于推送订阅，分发代理和合并代理在分发服务器上运行；对于请求订阅，则在订阅服务器上运行。

10.2 事　　务

在介绍 T-SQL 对事务的支持之前，先来介绍事务究竟是什么，在介绍了事务的基本概念之后，我们将讨论本地与分布式事务。

10.2.1 什么是事务

事务的概念是现代数据库理论的核心概念之一。事务是单个的工作单元。如果某一事务成功，则在该事务中进行的所有数据修改均会提交，成为数据库中的永久组成部分。如果事务遇到错误且必须取消或回滚，则所有数据修改均被清除。

SQL Server 以下列事务模式运行。

- 自动提交事务：每条单独的语句都是一个事务。
- 显式事务：每个事务均应以 BEGIN TRANSACTION 语句显式开始，以 COMMIT 或 ROLLBACK 语句显式结束。
- 隐式事务：在前一个事务完成时新事务即隐式启动，但每个事务仍以 COMMIT 或 ROLLBACK 语句显式完成。
- 批处理级事务：只能应用于多个活动结果集（MARS），在 MARS 会话中启动的 Transact-SQL 显式或隐式事务会变为批处理级事务。当批处理完成时，没有提交或回滚的批处理级事务自动由 SQL Server 进行回滚。

下面将举出几个例子帮助读者更好地理解事务的概念。

例如，通过 UPDATE 查询更新表行就被 SQL Server 作为单个事务来对待。假设执行下列查询：

```
USE db_stu
GO

UPDATE student
SET 系别='计算机系'
学号='20000008'
WHERE 姓名='李明'
```

当运行这个查询时，SQL Server 认为用户的意图是在单个行动中同时修改【系别】列和【学号】列。假设【学号】列上存在约束，使得【学号】列的更新无法实现。在这种情况下，【系别】列和【学号】列的更新都无法实现。由于这两个更新位于同一条 UPDATE 语句中，所以 SQL Server 将这两个更新作为同一个事务的一部分来对待。

如果希望这两个更新能被独立处理，则可以将上一条 UPDATE 语句改写成如下所示的两条 UPDATE 语句：

```
USE db_stu
GO

UPDATE student
```

```
SET  系别='计算机系'
WHERE  姓名='李明'

UPDATE student
SET  学号='20000008'
WHERE 姓名='李明'
```

经过改写之后，即使对【学号】列的更新没有成功，但是对【系别】列的更新仍能进行。

下面介绍使用 T-SQL 语句创建跨多条语句的事务。例如，可以执行下列批处理：

```
DECLARE @SSL_ERR INT，@RP_ERR INT
BEGIN TRANSACTION
UPDATE student
SET  系别='计算机系'
WHERE  姓名='李明'
SET @SSL_ERR=@@ERROR
UPDATE student
SET  学号='20000008'
WHERE  姓名='李明'
SET @RP_ERR=@@ERROR
IF @SSL_ERR=0 AND @RP_ERR=0
COMMIT    TRANSACTION
ELSE
ROLLBACK TRANSACTION
```

BEGIN TRANSACTION 语句通知 SQL Server，它应该将下一条 COMMIT TRANSACTION 语句或 ROLLBACK TRANSACTION 语句以前的所有事情作为单个事务。如果 SQL Server 遇到一条 COMMIT TRANSACTION 语句，那么保存自最近一条 BEGIN TRANSACTION 语句以后对数据库所做的所有工作；如果 SQL Server 遇到一条 ROLLBACK TRANSACTION 语句，则将抛弃所有这些工作。

10.2.2　ACID 属性

在形式上，我们可以说事务由 ACID 属性标示。ACID 是 4 个属性的首字母缩写词：原子性（Atomicity）、一致性（Consistency）、隔离性（Isolation）和持久性（Durability）。

事务是作为单个逻辑工作单元执行的一系列操作。一个逻辑工作单元必须有原子性、一致性、隔离性和持久性，只有这样才能成为一个事务。

1．原子性

原子性是谈及事物是工作单元这一概念的极富想象力的方法。当事务结束时，事务内的所有工作在数据库中要么都得到完成，要么都没有得到完成。数据库不能处于事务只有一部分得到完成的状态。

2．一致性

事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应

用于事务的修改，以保持所有数据的完整性。事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。

3．隔离性

由并发事务所做的修改必须与任何其他并发事务所做的修改隔离。事务识别数据时数据所处的状态，要么是另一并发事务修改它之前的状态，要么是修改它之后的状态，事务不会识别中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，并且重播一系列事务，以使数据结束时的状态与原始事务执行时的状态相同。

由于事务是相互隔离的，所以如果从数据库的一个崭新副本开始，并按最初执行所有操作的相同顺序再执行一遍它们，保证能得到相同的结果。这正是能够从备份与事物日志中还原事务的原因所在。

注意：从理论上说，一个事务绝不会遇到另一个事务内的状态，但实际上，SQL Server 允许通过选择隔离级别来改变这种情况。

最后，一旦提交了事务，它们就变成永久的了。事务所完成的工作会得到永久保存。如果提交一个事务以后计算机瘫痪，那么重新启动计算机后，该事务的结果将依然是存在的。

10.2.3 使用事务

Transact-SQL 使用下列4条语句管理事务：BEGIN TRANSACTION、COMMIT TRANSACTION、ROLLBACK TRANSACTION 和 SAVE TRANSACTION。

此外，还有两个全局变量可以用在事务处理中，即@@ERROR 和@@TRANCOUNT。

本节将逐一介绍这些语句的语法，以及如何在 T-SQL 批处理内使用事务处理。

1．BEGIN TRANSACTION 语句

BEGIN TRANSACTION 语句用来命令 SQL Server 开始一个新事物。

语法：

```
BEGIN{TRAN | TRANSACTION}
[{transaction_name | @tran_name_Variable}
[WITH MARK['description']
 ]
[;]
```

为了更好地理解上述语法，下面对参数进行说明。

- transaction _name：分配给事务的名称。transaction name 必须符合标志符规则，但标志符所包含的字符数不能大于 32。仅在最外面的 BEGIN…COMMIT 或 BEGIN…ROLLBACK 嵌套语句对中使用事务名。
- @tran _name_ variable：用户定义的、含有有效事务名称的变量的名称。必须用 char、varchar、nchar 或 nvarehar 数据类型声明变量。如果传递给该变量的字符多于 32 个，则仅使用前面的 32 个字符，其余的字符将被截断。
- WITH MARK【'description'】：指定在日志中标记事务。description 是描述该标记的字符串。如果 description 是 Unicode 字符串，那么在将长于 255 个字符的值存储到 msdb.dbo.1ogmarkhistory 表之前，先将其截断为 255 个字符。如果 description 为非 Unicode 字符串，则长于 510 个字符的值将被截断为 510 个字符。

注意：虽然事务名遵守 SQL Server 标志符的正常规则，但这些名称只有前 32 个字符有效。

可以使用 BEGIN、BEGIN TRAN 或 BEGIN TRANSACTION 作为基本语句。许多人更喜欢较短的形式，但长形式更容易理解。

BEGIN TRANSACTION 代表一点，由连接引用的数据在该点逻辑和物理上都是一致的。如果遇到错误，在 BEGIN TRANSACTION 之后的所有数据改动都能进行回滚，以将数据返回到已知的一致状态。每个事务继续执行，直到无误地完成并且用 COMMIT TRANSACTION 对数据库做永久的改动，或者遇上错误并且用 ROLLBACK TRANSACTION 语句擦除所有改动为止。

BEGIN TRANSACTION 为发出本语句的连接启动一个本地事务。根据当前事务隔离级别的设置，为支持该连接所发出的 Transact-SQL 语句而获取的许多资源被该事务锁定，直到使用 COMMIT TRANACTION 或 ROLLBACK TRANSACTION 语句完成该事务为止。长时间处于未完成状态的事务会阻止其他用户访问这些锁定的资源，也会阻止日志截断。

虽然 BEGIN TRANSACTION 启动了一个本地事务，但是在应用程序接下来执行一个必须记录的操作（如执行 INSERT、UPDATE 或 DELETE 语句）之前，它并不被记录在事务日志中。应用程序能执行一些操作，如为了保护 SELECT 语句的事务隔离级别而获取锁，但是直到应用程序执行一个修改操作后日志才有记录。

在一系列嵌套的事务中，用一个事务名给多个事务命名对该事务没有什么影响。系统仅登记第一个（最外部的）事务名。回滚到其他任何名称（有效的保存点名除外）都会产生错误。事实上，回滚之前执行的任何语句都不会在错误发生时回滚。这些语句仅当外层的事务回滚时才会进行回滚。

如果在语句提交或回滚之前执行了如下操作，那么 BEGIN TR ANSACTION 语句启动的本地事务将升级为分布式事务。

- 执行一个引用连接服务器上的远程表的 INSERT、DELETE 或 UPDATE 语句。如果用于访问链接服务器的 OLE DB 访问接口不支持 ITransactionJoin 接口，则 INSERT、UPDATE 或 DELETE 语句会失败。
- 当启用 REMOTE_PROC_TRANSACTIONS 时，将调用远程存储过程。

下面举一个例子来帮助读者更好地理解嵌套的事务。

```
BEGIN  TRANSACTION
UPDATE student
SET  系别='计算机系'
WHERE  姓名='李明'
BEGIN TRANSACTION
UPDATE student
SET  学号='20000008'
WHERE  姓名='李明'
COMMIT  TRANSACTION
ROLLBACK  TRANSACTION
```

在本例中，COMMIT TRANSACTION 语句通知 SQL Server，已经使用完自己启动的第二个事务。但是，ROLLBACK TRANSACTION 语句回退自第一条 BEGIN TRANSACTION 语句

以后的所有工作，其中包括内层嵌套事务。

事务可以是嵌套的，也就是说，可以在发布了一条 BEGIN TRANSACTION 命令之后发布另一条 BEGIM TRANSACTION 命令，然后或提交或回退等待处理的事务。这样就在第一个事务之内嵌套了第二个事务。原则是必须先提交或回退内层事务，然后再提交或回退外层事务，换句话说，一条 COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句对应最近的一条 BEGIN TRANSACTION 语句。

2．COMMIT TRANSACTION 语句

COMMIT TRANSACTION 语句的语法与 BEGIN TRANSACTION 语句非常相似。由于有同样的目的，也有一条替代语句：

```
COMNIT [TRAN | TRANSACTION][transaction_name|@tran_name_va rilable]
COMMIT[WORK]
```

为了更好地理解上述语法，下面对参数进行说明。

- Transaction_name：SQL Server Database Engine 忽略此参数。Transaction_ name 可指定由前面的 BEGIN TRANSACTION 分配的事务名称。Transaction_name 必须符合标志符规则，但不能超过 32 个字符。Transaction_name 通过向用户指明 COMMIT TRANSACTION 与哪些 BEGIN TRANSACTION 相关联，可作为帮助阅读的一种方法。
- @tran_name_variable：用户定义的、含有有效事务名称的变量的名称。必须用 char、varchar、nchar 或 nvarchar 数据类型声明变量。如果传递给该变量的字符数超过 32，则只使用前 32 个字符，其余的字符将被截断。

在发布一条 COMMIT TRANSACTION 语句时，SQL Server 将最近的一个已启动事务标记为准备提交。只有在提交一个嵌套事务系列中的最外层事务时，SQL Server 才将所有修改都写入到数据库中。当然，如果只有一个打开的事务，SQL Server 则立即将修改写入到数据库中。

用户的责任是保证在做完了所有预期的修改之后再发布 COMMIT TRANSACTION 语句。一旦提交了一个事务，就再也不回退它。

虽然可以在 COMMIT TRANSACTION 语句中使用事务名，但 SQL Server 并不尝试将这个名称与 BEGIN TRANSACTION 语句中的名称匹配。事务名的作用仅是增强代码的可读性。

COMMIT 语句（不管是否使用 WORK 关键字）仅仅是不带事务名的 COMMIT TRANSACTION 语句的同义词。这个语句形式与 ANSI SQL-92 标准是兼容的。

如果所提交的事务是 Transact-SQL 分布式事务，COMMIT TRANSACTION 将触发 MS DTC，使用两阶段提交协议，以便提交所有涉及该事务的服务器。如果本地事务跨越同一数据库引擎实例上的两个或多个数据库，则该实例将使用内部的两阶段提交协议来提交所有涉及该事务的数据库。

3．ROLLBACK TRANSACTION 语句

ROLLBACK TRANSACTION 语句也有两种形式：

```
ROLLBACK[ TRAN | TRANSACTION]
[Transaction_name |
@tran_name_variable |
Savepoint_name |
@savepoint_variable]
```

```
ROLLBACK[WORK]
```

为了更好地理解上述语法，下面对参数进行说明。

- Transaction_name：是为 BEGIN TRANSACTION 上的事务分配的名称。Transaction_name 必须符合标志符规则，但只使用事务名称的前 32 个字符。嵌套事务时，transaction_name 必须是最外面的 BEGIN TRANSACTION 语句中的名称。
- @tran_name_variable：是用户定义的、包含有效事务名称的变量的名称。必须用 char、varchar、nchar 或 nvarchar 数据类型声明变量。
- Savepoint_name 是 SAVE TRANSACTION 语句中的保存点的名称。Savepoint_name 必须符合标志符规则。当条件回滚只影响事务的一部分时，可使用 savepoint_name。
- @saveooint_variable：是用户定义的、包含有效保存点名称的变量的名称。必须用 char、varchar、nchar 或 nvarchar 数据类型声明变量。

ROLLBACK TRANSACTION 语句抛弃自最近一条 BEGIN TRANSACTION 语句以后的所有修改。同样，可以用常量或变量形式提供一个事务名，但 SQL Server 会将其忽略。

可以通过提供一个保存点名称来回退事务的一部分。如果事务是分布式事务（影响多个服务器上的数据库），则不能回退到保存点。

ROLLBACK（不管是否使用 WORK 关键字）是 ROLLBACK TRANSACTION 语句的 SQL-92 兼容形式。但是，ROLLBACK 无法回退嵌套事务系列中的单个事务。ROLLBACK WORK 总是回退到嵌套事务系列当中的第一个事务（最外层事务）。

在存储过程中，不带 savepoint_name 和 transaction_name 的 ROLLBACK TRANSACTION 语句将所有语句回滚到最外面的 BEGIN TRANSACTION。在存储过程中，ROLLBACK TRANSACTION 语句使@@TRANCOUNT 在触发器完成时的值不同于调用该存储过程时的@@TRANCOUNT 值，并且生成信息性消息。该信息不影响后面的处理。

如果在触发器中发出 ROLLBACK TRANSACTION：

- 将回滚对当前事务中的那一点所做的所有数据修改，包括触发器所做的修改；
- 触发器将继续执行 ROLLBACK 语句之后的所有其余语句，如果这些语句中的任意语句修改数据，则不回滚这些修改，执行其余的语句不会激发嵌套触发器；
- 在批处理中，不执行所有位于激发触发器的语句之后的语句。

当输入触发器时，@@TRANCOUNT 将以 1 递增，即使在自动提交模式下也是如此（系统将触发器视为隐含的嵌套事务处理）。

在存储过程中，ROLLBACK TRANSACTION 语句不影响调用该过程的批处理中的后续语句，将执行批处理中的后续语句。在触发器中，ROLLBACK TRANSACTION 语句终止包含激发触发器的语句的批处理，不执行批处理中的后续语句。

ROLLBACK TRANSACTION 语句不生成显示给用户的消息。如果在存储过程或触发器中需要警告，应使用 RAISERROR 或 PRINT 语句。RAISERROR 是用于指出错误的首选语句。

ROLLBACK 对游标的影响由下面 3 个规则定义。

- 当 CURSOR_CLOSE_ON_COMMIT 设置为 ON 时，ROLLBACK 关闭但不释放所有打开的游标。
- 当 CURSOR_CLOSE_ON_COMMIT 设置为 OFF 时，ROLLBACK 不影响任何打开的同步 STATIC 或 INSENSITIVE 游标，也不影响已完全填充的异步 STATIC 游标，它将关闭但不释放任何其他类型的打开的游标。

- 终止批处理并生成内部回滚的错误将释放在包含错误声明的批处理中声明的所有游标，而不考虑它们的类型或 CURSOR_CLOSE_ON_COMMIT 的设置情况。这包括在错误批处理调用的存储过程中声明的游标。在错误批处理之前的批处理中声明的游标遵循前两个规则，死锁错误便是此类错误的一个示例，在触发器中发出的 ROLLBACK 语句也将自动生成此类错误。

4．SAVE TRANSACTION 语句

SAVE TRANSACTION 语句允许部分地提交一个事务，同时仍能回退这个事务的其余部分。

语法：

```
SAVE{TRAN | TRANSACTION }{savepoint_name | @savepoint_variable}
```

为了更好地理解上述语法，下面对参数进行说明。

- savepoint_name：分配给保存点的名称。保存点名称必须符合标志符的规则，但长度不能超过 32 个字符。
- @savepoint_variable：包含有效保存点名称的用户定义变量的名称。必须用 char、varchar、nchar 或 nvarchar 数据类型声明变量。如果长度超过 32 个字符，也可以传递到变量，但只使用前 32 个字符。

用户可以在事务内设置保存点或标记。保存点可以定义在按条件取消某个事务的一部分后，该事务可以返回的一个位置。如果将事务回滚到保存点，则根据需要必须完成其他剩余的 Transact-SQL 语句和 COMMIT TRANSACTION 语句，或者必须通过将事务回滚到起始点完全取消事务。若要取消整个事务，应使用 ROLLBACK TRANSACTION transaction_name 语句，这将撤销事务的所有语句和过程。

在事务中允许有重复的保存点名称，但指定保存点名称的 ROLLBACK TRANSACTION 语句只将事务回滚到使用该名称的最近的 SAVE TRANSACTION。

注意：在发布 SAVE TRANSACTION 时，必须给事务提供一个名称。这个名称将为一条后续的 COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句提供参照点。

当事务开始后，事务处理期间使用的资源将一直保留，直到事务完成（也就是锁定）。当将事务的一部分回滚到保存点时，将继续保留资源直到事务完成（或者回滚整个事务）。

下面举例说明 SAVE TRANSACTION 语句的用法。

```
BEGIN TRANSACTION
UPDATE    student
SET  系别='计算机系'
WHERE    姓名='李明'
SAVE TRANSACTION SSsAVED
UPDATE student
SET    年龄='26'
WHERE    姓名='李明'
ROLLBACK TRANSACTION SSsAVED
COMMIT TRANSACTION
```

假设，【年龄】列的约束使得其值必须小于【25】。在这种情况下，ROLLBACK TRANSACTION 语句删除对【年龄】列的更新结果，同时保留对【系别】列的更新，以备提

交。然后，COMMIT TRANSACTION 语句提交事务中没有回退的部分。

5．@@TRANCOUNT 变量

@@TRANCOUNT 是系统全局变量，用来报告当前等待处理的嵌套事务数量。如果没有等待处理的事务，这个变量则包含 0。例如，这个变量适用于确定在 T-SQL 批处理所启动的一个事务中是否有正在执法的触发器。

BEGIN TRANSACTION 语句将@@TRANCOUNT 加 1。ROLLBACK TRANSACTION 将@@TRANCOUNT 递减到 0，但 ROLLBACK TRANSACTION Savepoint_name 除外，它并不影响@@TRANCOUNT。COMMIT TRANSACTION 或 COMMIT WORK 将@@TRANCOUNT 递减 1。为了更好地帮助读者理解@@TRANCOUNT 变量的含义及其用法，下面我们举一个例子供读者学习。

以下示例使用@@TRANCOUNT 测试应该提交的打开事务：

```
USE db_stu
GO
BEGIN TRANSACTION
GO
UPDATE student
SET 系别='计算机系'，年龄='26'
WHERE 姓名='李明'
GO
IF @@TRANCOUNT>0
BEGIN
PRINT 'A Transaction needs to be rolled back.';
Rollback transaction
END
```

6．@@ERROR 变量

@@ERROR 变量是系统全局变量，用来保存当前来自任何一条 T-SQL 语句的最新错误号。每当一条不引起错误的语句执行完毕时，这个变量则包含 0。换句话说，每当一条语句成功地执行完毕时，该变量就复位到 0。因此，如果需要在以后的某个时候检查语句是否引起错误，则需要@@ERROR 变量的值先保存到局部变量中。

下面我们举一个例子帮助读者更好地理解@@ERROR 变量。

以下示例使用@@ERROR 在 UPDATE 语句中检测约束检查冲突：

```
USE db_stu
GO
BEGIN TRANSACTION
GO
UPDATE student
SET 系别='计算机系'，年龄='cccc'
WHERE 姓名='李明'
GO
```

```
 IF @@ERROR=293
PRINT 'A check constraint violation occurred.'
GO
```

10.2.4　事务的举例

本节用一个比较复杂的 T-SQL 批处理演示以上所介绍的各种事务处理语句，以便帮助读者更好地掌握事务应用。具体示例如下:

```
USE db_stu
GO
DECLARE@SS_ERR INT，@RP_ERR INT
BEGIN TRANSACTION
UPDATE student
SET 系别='计算机系'
WHERE 姓名='李明'
SET @SS_ERR=@@ERROR
SAVE TRANSACTION SSsaved

UPDATE student
SET 年龄='26'
WHERE 姓名='李明'
SET @RP_ERR=@@ERROR
IF @RP_ERR<>0
ROLLBACK TRANSACTION SSsaved
IF @SS_ERR=0 AND @RP_ERR=0
BEGIN
COMMIT TRANSACTION
PRINT'CHANGES WERE SUCCESSFUL'
END
ELSE
ROLLBACK TRANSACTION
```

下面逐一说明这个批处理的每个组成部分。

- DECLARE 语句设立两个局部变量。
- BEGIN TRANSACTION 语句开始一个事务。
- 第一条 UPDATE 语句对“系别”列做一项修改。
- 第一条 SET 语句用来保护@@ERROR 变量的值，以便以后能检查第一条 UPDATE 语句是否执行顺利。需要注意的是，这条语句必须紧跟在 UPDATE 语句的后面。
- SAVE TRANSACTION 语句设置一个保存点。
- 第二条 UPDATE 语句对“年龄”列做一项修改。
- 第二条 SET 语句保存@@ERROR 变量的值，以便以后能够检查该 UPDATE 语句是否

执行顺利。

- 如果第二条 UPDATE 语句上发生错误，则第一条 ROLLBACK TRANSACTION 语句将事务退回到保存点。
- 如果没有错误，则事务得到提交，一个消息打印出来。请注意 BEGIN 与 END 的使用，它们用来将两条 T-SQL 语句分组到同一条逻辑语句中。这是必要的，因为在默认情况下，IF 语句只引用紧跟在它后面的那条语句。
- 如果发生错误，则第二条 ROLLBACK TRANSACTION 语句回退所有工作。

10.2.5　分布式事务

到目前为止，我们一直在讨论本地事务，即仅在单个数据库中做修改的事务。SQL Server 还支持分布式事务，即对多个数据库中存在的数据做修改的事务。这些数据库不必是 SQL Serve 数据库，也可以是其他连接到服务器上的数据库。

在分布式数据库系统中，一个全局事务会涉及多个局部数据库上的数据更新。因此，可以把一个分布事务看成在不同的局部事务中执行的子事务的集合。分布事务要求：组成该事务的所有子事务要么一起提交，要么一起回滚。保证分布事务的 ACID 特性复杂得多。

在分布式数据库系统中，通常每个数据库节点都有一个局部事务管理器，SQL Server 使用分布式事务处理协调器（MS DTC）作为局部事务管理器。MS DTC 服务用来管理子事务的执行，保证子事务的完整性。同时，这些局部事务管理器还必须相互协调，保证所有节点服务器的子事务要么都提交，要么都回滚，从而保证全局事务的原子性。

分布式事务可以用代码来管理，代码中使用的 SQL 语句跟用来管理本地事务的 SQL 语句完全相同。但是，在分布式事务中发布 COMMIT TRANSACTION 语句时，SQL Server 会自动调用一个称为两步提交的协议，也称为两阶段提交（2PC）。

- 准备阶段：当事务管理器收到提交请求时，它会向该事务涉及的所有资源管理器发送准备命令。然后，每个资源管理器将尽力使该事务持久，并且所有保存该事务日志映像的缓冲区将被刷新到磁盘中。当每个资源管理器完成准备阶段时，它会向事务管理器返回准备成功或准备失败的消息。
- 提交阶段：如果事务管理器从所有资源管理器收到准备成功的消息，它将向每个资源管理器发送一个提交命令。然后，资源管理器就可以完成提交了。如果所有资源管理器都报告提交成功，那么事务管理器就会向应用程序发送一个成功通知。如果任一资源管理器报告准备失败，那么事务管理器将向每个资源管理器发送一个回滚命令，并向应用程序表明提交失败。

1．Microsoft DTC 服务

分布式事务由一个称为分布式事务协调程序（Distributed Transaction Coordinator，DTC）的 SQL Server 构件来管理。这是一个独立服务，在 SQL Server 安装期间安装。如果打算使用分布式事务，则应当将这个事务设置为自动启动。

2．BEGIN DISTRIBUTED TRNSCTION 语句

BEGIN DISTRIBUTED TRANSACTION 语句用来显式地通知 SQL Server 启动分布式事务。

语法：

```
BEGIN DISTRIBUTED{TRAN|TRANSACTION}
```

```
    [transaction_name | @tran_name_variable]
```

为了更好地理解上述语法，下面对参数进行说明。

- transaction_name：用户定义的事务名，用于跟踪 MS DTC 实用工具中的分布式事务。transaction_name 必须符合标志符规则，字符数必须≤32。
- @tran_name_variable：用户定义的一个变量名，它含有一个事务名，该事务名用于跟踪 MS DTC 实用工具中的分布式事务。必须用 char、varchar、nchar 或 nvarchar 数据类型声明变量。

这条语句有别于普通 BEGIN TRANSACTION 语句的唯一地方是包含 DISTRIBUTED 关键字。

如果在事务期间修改远程服务器上的数据，本地事务则自动上升为分布式事务。例如，如果在远程服务器上执行 INSERT、UPDATE 或 DELETE 语句，或者调用远程存储过程，同时又正处于事务的中途，那么这个事务将变成分布式事务。

执行 BEGIN DISTRIBUTED TRANSACTION 语句的 SQL Server Database Engine 的实例是事务创建者，并控制事务的完成。当为会话发出后续 COMMIT TRANSACTION 或 ROLLBACK TRANSACTION 语句时，控制实例请求 MS DTC 在所涉及的所有实例间管理分布式事务的完成。

3．事务提示

事务要占用服务器上的资源。具体地说，当用户修改事务中的数据时，这个数据必须被锁定，以确保它在事务提交完毕时是可用的。因此，一般说来，应当让事务保持在有效状态，以避免给其他用户带来麻烦。下面是一些要考虑的要点。

- 不要在事务内做任何一件要求用户交互的事情，因为这会在应用程序等待用户输入期间引起长时间的锁定。
- 不要为单条 SQL 语句使用一个事务。
- 在事务期间修改尽可能少的数据。
- 不要在用户正浏览数据时启动事务，应当等到用户准备实际修改数据时再启动事务。
- 应当使事务尽可能的短。

10.3　自动化管理基础

所谓自动化管理实际上是对预先已经预测到的服务器事件或按时必须执行的管理任务，根据已经订好的计划做出必要的反应。通过自动化管理，用户可以将一些每天都必须进行的固定不变的日常维护任务交给服务器自动执行。当服务器发生异常事件时，自动发出通知，以便让操作人员及时获得信息，并进行及时处理。

10.3.1　自动化管理概述

在 SQL Server 2008 中很多项管理任务都可以设置成自动化来实现。这些管理任务主要包括以下方面：

- 任何 Transact-SQL 语法中的语句；
- 操作系统命令；
- VBScript 或 JavaScript 之类的脚本语言；
- 复制任务；
- 数据库创建和备份；

- 索引重构；
- 报表生成。

由此可以看出 SQL Server 的自动化功能非常强大，但是要实现自动化管理，通常需要管理员预先完成以下工作：

- 找出可能会周期性出现的管理任务或服务器事件，从中筛选出可以预先提出解决方案的任务或事件；
- 定义一系列的作业和警报；
- 合理配置并运行 SQL Server 代理服务。

当要跨多个服务器进行自动化管理时称为“多服务器管理”。在多服务器管理时必须至少有一台主服务器且至少有一台目标服务器。主服务器将作业分发到目标服务器，并从它那里接收事件。主服务器还存储在目标服务器上运行的作业定义的中央副本中。目标服务器定期连接到主服务器来更新它们的作业计划。如果主服务器上存在新作业，目标服务器将下载该作业。目标服务器在完成作业后，会重新连接到主服务器并报告作业状态。

多服务器自动化管理可以在一定程度上大大减轻服务器管理员的工作量。如果用户是一个大型数据库备份管理员，则可以在主服务器上定义一个备份作业，并制定该作业的执行时间表。这样，所有的目标服务器都将自动从主服务器上下载该作业，并自动按时间表的规定完成该作业。这样，用户虽然只做了一次作业的定义，却可以完成范围较大的备份任务。

通常情况下，多服务管理的任务应该由具有 sysadmin 固定服务器角色的用户来完成。但是，目标服务器上的 sysadmin 固定服务器角色成员无法编辑修改由主服务器在目标服务器上执行的操作，这种机制可以避免目标服务器的用户在不知情的情况下误删除作业而导致服务器不能正常运行。

10.3.2　自动化管理元素

SQL Server 的自动化能力的核心是 SQL Server 代理服务。自动化和复制是这个服务特有的两大功能。这个服务能使警报、操作员和作业完成它们的自动化功能。

1．作业

作业是定义自动任务的一系列步骤。用户可以使用作业来定义将要执行一次或多次的管理任务，并监督该任务的完成情况。作业可以在本地服务器和多台远程服务器上运行，可以按一定的时间表运行，也可以通过警报来触发执行。

2．警报

警报是 SQL Server 中产生并记录在 Windows 应用程序日志中的错误消息或事件。它可通过电子邮件、传呼机或 Net Send 发送给用户。如果错误消息没有记录在 Windows 应用程序日志中，警报则无法激活。

3．操作员

当警报激活时，它们可以发送给用户。需要接收这些消息的用户在 SQL Server 中称为操作员。操作员用来配置谁接收警报和他们何时可以接收警报。操作员可以是一个用户，也可以是多个用户。

其实，SQL Server 对传呼的事情一无所知。传呼通知方式必须与电子邮件服务紧密结合。通常情况下，邮件服务器上要安装能访问寻呼台的网关服务。

下面举例说明这 3 个元素如何合作完成管理的自动化。

假设李华是一个 SQL Server 的服务器管理人员，负责每天备份两台通过网络连接的 SQL

Server 服务器上的数据。李华希望备份工作能准确地完成，当备份执行过程中出现问题时，服务器应该能够及时通知他，以便能迅速解决问题。

为了能自动完成每天的备份工作，李华设计了以下操作。

（1）为了取得操作权限，设置李华为 DailybackupOperator 操作员。

（2）定义一台服务器为主服务器，而另一台为目标服务器。

（3）定义一个备份作业，并指定该作业在每天晚上 22 点自动执行。

（4）当该作业运行发生错误时，将一条错误信息写到 Windows 事件日志上。

（5）当 SQL Server 代理服务读取 Windows 事件日志时，这个代理发现了失败的作业所写入的错误信息，并将其与 MSDB 数据库中的 sysalerts 表进行比较。

（6）当代理找到一个匹配项目时，激活一个警报。

（7）该警报在激活时就会发送电子邮件通知李华。

（8）李华及时对问题进行解决。

但是，要让这个作业的任何一部分正常工作，SQL Server 代理服务必须配置正确，即这个代理必须正常运行才能使自动化成为可能。我们可以通过 3 种方式查看 SQL Server 代理服务是否正常运行。首先，可以打开 SQL Server Management Studio，并查看【SQL Server 代理】图标，如果该图标是内有 X 的红色圆环，则该服务已经停止；如果它是个绿色箭头，则该服务开启。其次，可以通过右键菜单选择相应命令来启动或停止 SQL Server 代理服务。最后，还可以使用【计算机管理】或【控制面板】中的【服务】工具，来检查和改变该服务的状态。

在代理服务运行正常的情况下，最好使用一个域账户而不是本地系统账户登录，因为使用本地系统账户将不允许使用网上的其他 SQL Server。换句话说，将无法执行多服务器作业完成复制或使用 SQL Server 的电子邮件功能。要确定这个代理是不是用域账户登录的，可打开【控制面板】，然后打开【管理工具】文件夹，再打开【服务】工具，找到 SQL Server Agent 并双击，打开 SQL Server 代理属性窗口，选择【登录】选项卡，如图 10.1 所示。

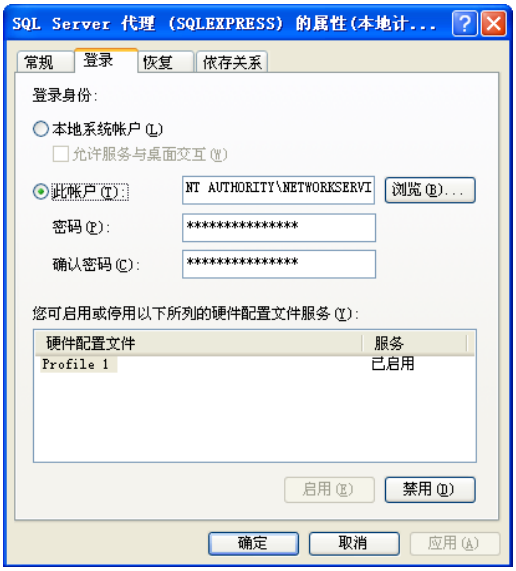


图 10-1　【登录】选项卡页面

选中【此账户】单选按钮，单击【浏览】按钮，打开【选择用户】对话框，选择相应的域

用户即可。

注意：要实现 SQL Server 自动化管理，必须确保 SQL Server 代理服务正在运行，所以建议用户可以将 SQL Server 代理设置为操作系统启动时自动启动。

10.4　配置数据库邮件

SQL Server 最突出的管理性能之一在于能够将服务器与邮件系统集成起来。一旦配置好“数据库邮件”以后，就可以使用该邮件系统来处理警报通知。

10.4.1　数据库邮件概述

“数据库邮件”是 SQL Server 2005 版本以后的一个新增服务，用来代替 SQL Server 服务发送电子邮件。SQL Server 的早期版本含有 SQL Mail 和 SQL Agent Mail 两个服务，这两个服务均使用 MAPI 并要求服务器上装有 MAPI 客户端（通常是 Outlook）。“数据库邮件”不使用 MAPI，而是使用标准的简单邮件传输协议（Simple Mail Transfer Protocol，SMTP），因此不要求安装 MAPI 客户端。

使用“数据库邮件”有以下优点。

- 这个处理邮件的应用程序（DatabaseMail90.exe）运行时为一个独立的进程，因此，如果出了问题，SQL Server 不受影响。
- 可以指定多个邮件服务器，如果其中的一个邮件服务器出现问题，“数据库邮件”仍然能够处理邮件。

“数据库邮件”具有可伸缩性，因为它在后台处理邮件。当创建一个发送邮件的请求时，“数据库邮件”给 Service Broker 队列添加一个请求。Service Broker 队列允许这个请求以异步方式处理，如果服务器在能处理它以前已经停机，那么将保存这个请求。

- “数据库邮件”的多个副本可以同时运行，而且在同一个服务器上可以有多个邮件配置文件和邮件主数据库。
- “数据库邮件”比它的前任更安全、更容易管理。它有精细的控制，因此可以限定哪些用户可以发送邮件。另外，可以指定允许和禁止哪些文件扩展名作为附件，以及附件的最大长度。
- “数据库邮件”所做的每件事情都记录在 Windows 应用程序日志中，而发出的消息仍保留在邮件主机数据库中以供审核。

10.4.2　配置数据库邮件过程

在开始配置“数据库邮件”之前，网络上的某个地方应当有一个 SMTP 邮件服务器，并且该服务器有一个针对 SQL Server Agent 服务账户而配置的邮件账户。具体如何安装和配置 SMTP 服务器不在本书介绍的范围内，如有需要可参考其他相关书籍。但是，如果已经向某个因特网服务提供商（ISP）注册了一个电子邮件账户，则可以使用这个账户，并通过配置向导配置“数据库邮件”。下面是配置一个邮件主机数据库的过程。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接到服务器。

（2）在【对象资源管理器】窗口中，展开【管理】节点，右击【数据库邮件】节点，在菜

单中选择【配置数据库邮件】命令。打开【数据库邮件配置向导】欢迎窗口。

（3）在【数据库邮件配置向导】窗口中，单击【下一步】按钮。打开【数据库邮件配置向导】的【选择配置任务】窗口。

（4）在【数据库邮件配置向导】的【选择配置任务】窗口中，选中【通过执行以下任务来安装数据库邮件】单选按钮，然后单击【下一步】按钮，系统将弹出一个对话框。

（5）单击【是】按钮启动“数据库邮件”功能，打开【数据库邮件配置向导】的【新建配置文件】窗口。

（6）在【数据库邮件配置向导】的【新建配置文件】窗口中的【配置文件名】文本框中输入 SQLMailConfigProfile，然后单击【添加】按钮，打开【新建数据库邮件账户】对话框，并输入相关内容。

（7）在【新建数据库邮件账户】对话框中，如果电子邮件服务器要求登录，则应选中【身份验证】单选按钮，并输入相关的用户名和密码，或者根据需要选中【使用数据库引擎服务凭据的 Windows 身份验证】或【匿名身份验证】单选按钮。

（8）设置完成相关内容后，单击【确定】按钮返回【数据库邮件配置向导】窗口。

（9）单击【下一步】按钮，打开【数据库邮件配置向导】的【管理配置文件安全性】窗口。

（10）在【数据库邮件配置向导】的【管理配置文件安全性】窗口上，选中刚才创建的邮件配置文件前面的【公共】复选框，用于让所有用户都可以访问它，并设置【默认配置文件】选项，然后单击【下一步】按钮，打开【数据库邮件配置向导】的【配置系统参数】窗口。

（11）在【数据库邮件配置向导】的【配置系统参数】窗口中，可以根据实际要求进行参数更改，或者接受默认设置，并单击【下一步】按钮，打开【数据库邮件配置向导】的【完成该向导】窗口。

（12）单击【完成】按钮，打开【数据库配置向导】的【正在配置】窗口。

（13）当系统配置完成“数据库邮件”后，单击【关闭】按钮完成“数据库邮件”配置。

10.4.3　使用邮件配置文件

现在，我们将 SQL Server Agent 服务配置成使用刚才创建的邮件配置文件。

（1）在【对象资源管理器】窗口中，右击【SQL Server 代理】节点，并从弹出的菜单中选择【属性】命令，打开【SQL Server 代理属性】窗口。

（2）打开【警报系统】选项窗口，选中【启用邮件配置文件】复选框，从【邮件系统】下拉列表中选择【数据库邮件】选项，从【邮件配置文件】下拉列表中选择【SQLMailConfigpProfile】选项。

（3）单击【确定】按钮完成属性设置。

（4）从【计算机管理】窗口中，停止并重新启动 SQL Server Agent 服务。

（5）在“数据库邮件”配置好以后，可以根据具体情况再次运行配置向导为配置进行修改。在配置向导中可以进行如下操作：

- 添加、删除账户或配置文件；
- 通过将配置文件标记为“公共”或“专用”来管理配置文件安全性；
- 浏览或修改系统参数；
- 卸载“数据库邮件”。

在顺利地配置了“数据库邮件”之后，就可以创建从 SQL Server 那里接收电子邮件的操作员了。

提示：“Internet 信息服务”携带了一个内部的 SMTP 服务器，该服务器与“数据库邮件”一起使用。

10.5 操　作　员

为了 SQL Server 出现问题时能够联系管理员，需要进行几个设置，包括跟谁联系、联系人何时可以接收信息、应该如何跟联系人进行联系（通过电子邮件、传呼机还是 Net Send），以及应该将哪些问题通知给哪些联系人。Net Send 消息指的是从源计算机发送给目标计算机的消息，它以对话框的形式突然出现在用户屏幕上，并位于所有打开应用程序的最前面。操作员就是 SQL Server 中用来配置这些设置的对象。

例如，假设公司中有几个人需要在 SQL Server 遇到问题时得到通知，并且需要以不同方式接到不同的问题警报。数据库管理员可能需要通过电子邮件接到任何数据库管理方面的问题（比如备份故障）和警报，开发人员需要通过传呼机接收编程方面（比如死锁）的警报，公司经理可能需要知道其他非技术性问题（比如删除或更改一些客户信息）的警报，为了实现这些要求，只要为每种类型的用户分别创建不同的操作员并配置相应的设置，就可以处理这些类型的用户了。

下面，我们将通过配置一个操作员来介绍具体的操作步骤。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQLServer 身份验证连接服务器。

（2）在【对象资源管理器】窗口中，展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【操作员】节点，在菜单中选择【新建操作员】命令，打开【新建操作员】窗口。

（4）在【名称】文本框中，输入 Administrator。

（5）如果已经将系统配置成使用 “数据库邮件”发送邮件，则输入电子邮件名称（这里输入 xieke@cybertang.com）即可；如果没有将系统配置成使用电子邮件，则跳过这一步。

（6）在 Net Send 文本框中，输入计算机名称，这里输入 computer。

（7）如果操作员携带了能够接收电子邮件的传呼机，则可以在【寻呼电子邮件名称】文本框中输入传呼机的电子邮件地址，这时输入 xieke@cybertang.com。

（8）在【寻呼值班计划】中，可以选择这个操作员可以接收通知的日期和时间，如果复选了某一天，操作员将在那一天的某个时间段（【工作日开始】和【工作日结束】选项指定的时间内）接到通知。这里保留默认值。

（9）单击【确定】按钮，完成操作员的新建。

由于种种原因，也许会出现某一时段无人值班的情况，那么如果那个时段出现了错误，则将没有操作员会接到警报，可能会产生意想不到的后果。为了避免这种情况的发生，应该为系统创建一个全故障保险操作员，此人负责接收无人值班时的警报。下面是创建故障保险操作员的步骤。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接服务器。

（2）在【对象资源管理器】窗口中，展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【SQL Server 代理】节点，从菜单中选择【属性】命令，打开【SQL Server 代理属性】窗口，选择【警报系统】选项，打开【警报系统】选项窗口。

（4）选中【启用防故障操作员】复选框；在【操作员】下拉列表中选择 Administrator 选项；在通知方式中，复选【Net Send】复选框，使用操作员以故障保险操作员的身份接收 Net Send 消息。

（5）单击【确定】按钮完成故障保险操作员的创建。

在创建了操作员之后，就可以开始创建自动化任务的作业了。

10.6　警　　报

如果没有警报发出通知，操作员将无法发挥出应有的作用。当在 SQL Server 上发生事件时，就会立即激活警报，然后这些警报可以发送给操作员，以使他们根据情况采取相应的措施。警报基于下列 3 个元素：错误号、错误严重级别和性能计算器。

SQL Server 中可以出现的错误都有编号（约 3 000 个）。即使已经列出了这么多种错误，但仍然不够。例如，假设希望在用户从客户数据库中删除客户时激活某个警报，但 SQL Server 并没有包括与数据库的结构或用户的名称有关的警报，因此需要创建新的错误号，并针对这样的私有事件产生一个警报。警报可以基于任何一个有效的错误号创建。

SQL Server 中的每个错误还有一个关联的严重级别，用于指示错误的严重程度。警报可以按严重级别产生。如下所述列出了比较常见的严重级别。

常见错误级别如下。

- 10：这是信息性消息，由用户输入信息中的错误所引起，它是不严重的。
- 11～16：这些是用户能够纠正的所有错误。
- 17：这些错误是在服务器耗尽资源（比如内存或硬盘空间）时产生的错误。
- 18：一个非致命的内部错误已经产生。语句将完成，并且用户连接将维持。
- 19：一个不可配置的内部限额已被达到。产生这个错误的任何语句都将被终止。
- 20：当前数据库中的一个单独进程已遇到问题，但数据库本身未造成破坏。
- 21：当前数据库中的所有进程都受到该问题影响，但数据库本身未遭到破坏。
- 22：正在使用的表或索引可能受到损坏。应该运行 DBCC 设法修复对象（问题也可能出在数据缓存中，也就是说，一个简单的重启可能就解决了问题）。
- 23：这条消息通常指整个数据库不知何故已遭破坏，而且应该检查硬件的完整性。
- 24：硬件已经发生故障。可能需要购买新硬件并从备份中重装数据库。

警报也可以从性能计数器中产生。这些计数器与性能监视器中的计数器完全相同，而且对纠正事务日志填满（或几乎填满）之类的性能问题是非常有用的。它也可以产生基于 WMI（Windows Management Instrumentation）事件的警报。下面我们介绍一下如何创建一些警报。

10.6.1　标准事件警报

标准警报是基于 SQL Server 中的内部错误消息与严重级别的警报。要创建基于这些事件之一的警报，必须将错误写到 Windows 事件日志上，因为 SQL Server 代理是从该事件日志上

读取错误信息的。一旦 SQL Server 代理读取了该事件日志并检测到了新错误，它就会搜索整个数据库查找匹配的警报。当这个代理发现匹配的警报时，该警报立即洗涤，进而可以通知操作员，执行作业或者同时做这两件事情。

假设，我们创建一个从错误中激活的警报（激活严重级别的警报也是如此，只是基于错误的严重级别，而不是错误的编号），在这里使用专门的警报激活命令 RAISERROR()。下面，我们创建一个基于错误号 1 的警报，它将发送一个 Net Send 通知给操作员。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接到服务器。

（2）在【对象资源管理器】窗口中，展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【警报】节点，从弹出的菜单中选择【新建警报】命令，打开【新建警报】窗口。

（4）在【名称】文本框中，输入 Alert 1；从【类型】下拉列表中选择【SQL Server 事件】选项；从【数据库名称】下拉列表中选择【所有数据库】选项；由于无法手工激活 13 000 号以下的错误，因此我们使用错误号 14 623，所以选中【错误号】单选按钮，在文本框中输入 14 623。

（5）选择【响应】选项，打开【响应】选项窗口，选中【通知操作员】复选框，并选中 Administrator 行的【Net Send】复选框。

（6）选择【选项】选项，打开【选项】选项窗口，选中【警报错误文本发送方式】下面的【Net Send】复选框。

（7）单击【确定】按钮，完成标准警报的创建。

上面设计了一个每当出现错误号 14 623 时就激活的警报，下面就用 RASIERROR()命令产生错误号 14 623。

首先，在【SQL Server Management Studio】中单击【新建查询】按钮，打开一个新的 SQL Server 查询窗口。

然后，输入并执行下列代码来激活这个错误：

```
RAISERROR(14623,10,1)
```

最后，当 Net Send 消息弹出时，仔细观察它给出的细节，其中包括错误号、描述和其他文本，然后单击【确定】按钮返回。

下面来分析一下这个过程。首先，创建了一个基于错误号 14 623 的警报；但是，由于这个错误号最初没有配置成写到 Windows 日志上，所以需要修改它，将它写到该事件日志上（如果错误不写到 Windows 事件日志中，则它的警报将永远不能被激活）。然后，将警报配置成每当被它激活时，都通过 Net Send 消息通知操作员。最后，使用 RAISERROR()命令强制该警报激活并发送通知。

其实许多警报都是由于用最少的 Transact-SQL 代码就可修复的问题而激活的。因此，可以将警报配置成执行一些作业（关于作业的创建与使用，将在 10.7 节进行详细的介绍），让那些作业去修复引起警报激活的问题。下面，我们将一个现有的警报修改成执行现有的作业。

（1）在【SQL Server Manager Studio】中，展开【SQL Server 代理】下面的【警报】节点。

（2）右击【Alert 1】节点，在弹出的菜单中选择【属性】命令，打开【“Alert 1”警报属性】窗口。

（3）选择【响应】选项，打开【响应】选项窗口，选中【执行作业】复选框，并在文本框中输入 Test。

（4）单击【确定】按钮应用修改结果。

（5）修改了警报后，再次用上面的方法激活该警报，并观察 Test 作业运行的情况。

从上面的创建过程中可以看出，创建基于内部错误的警报不是那么困难。虽然 SQL Server 有接近 3 700 种这样的错误，但仍然不能满足全部需要。因此，需要知道如何创建自定义的错误消息。

10.6.2　自定义事件警报

虽然 SQL Server 包含的错误种类很多，但是它并没有涵盖全部的情况。例如，如果客户用信用卡订购产品，那么管理人员就需要跟踪客户的信用情况。每当信用好的客户被删除或该客户信用额度被降低时，销售经理可能需要得到通知，用于了解公司当前的销售情况。但是 SQL Server 中没有包含这类默认错误的消息，所以必须创建一条这样的错误消息，然后才能用它激活该警报。

SQL Sever 允许创建任意多个错误类型，但错误号必须是从 50 001 开始的（这是所有用户定义错误的开始号）。下面，我们来创建一个用户自定义错误的警报，并利用该警报。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接服务器。

（2）在【SQL Server Management Studio】中，单击【新建查询】按钮，打开一个新的查询窗口。

（3）输入并执行下面的语句来创建这个新错误：

```
USE db_stu
GO
EXEC sp_addmessage @msgnum=50005,@severity=11
@msgtext=N'This is a custom error. ', @with_log='true'
```

（4）在【对象资源管理器】窗口中，展开【服务器】节点，然后展开【SQL server 代理】节点。

（5）右击【警报】节点，从弹出的菜单中选择【新建警报】命令，打开【新建警报】窗口，在【名称】文本框中输入 Customer Alert，选中【错误号】单选按钮，并在文本框中输入 50 005。

（6）选择【响应】选项，打开【响应】窗口，选中【通知操作员】复选框，并选中 Administrator 行的【Net Send】复选框。

（7）选择【选项】选项，打开【选项】选项窗口，选中【警报错误文本发送方式】下面的【Net Send】复选框。

（8）单击【确定】按钮，完成新警报的创建。

有了基于自定义错误消息的警报之后，就可以使用 RAISERROR()命令测试该警报了，具体过程如下。

（1）在【SQL Server Management Studio】中单击【新建查询】按钮，打开一个新的 SQL Server 查询窗口。

（2）输入并执行下面的语句来激活这个错误：

```
RAISERROR(50005，10，1)
```

（3）当 Net Send 消息弹出时，注意它给出的细节，然后单击【确定】按钮返回。

其实，使用上面的方法创建的警报虽然有效，但不太明确问题所在。所以如果我们需要了解更详细的情况，可以在用户定义的错误消息中使用参数来达到目的。

参数是在错误被激活时用来提供给信息的占位符。例如，指出“某个客户已经被删除”的消息在同一个错误每次被激活时都始终显示同一段静态文本，但是，如果在消息中使用参数，那结果就不一样了。可以用在消息中的参数如下：

- %ls 与%s 代表字符串；
- %ld 与%d 代表数字。

下面，举例说明参数是如何在消息中使用的。

（1）在【SQL Server Management Studio】中，单击【新建查询】按钮打开一个新的 SQL Server 查询窗口。

（2）输入并执行下面的语句来创建这个新错误：

```
GO
EXEC sp_addmessage@msgnum=50005，@severity=11
@msgtext=N'This is a custom error by %ls．',@with_log='TRUE'，
@replace='replace'
```

（3）输入并执行下面语句来激活这个错误：

```
RAISERROR(50005，10，1，'JackLong')
```

（4）当 Net Send 消息弹出时，可以注意到文本中已经包含了“JackLong”字符串，代替消息文本中的%ls，单击【确定】按钮返回。

所以要想让得到的 Net Send 消息能较清楚地表明错误的原因，可以在消息中使用参数。

10.6.3　性能警报

事件警报适合处理事后的问题，但并不是所有问题都可以等到出现之后再解决。有些问题要提前发现，以避免对系统造成破坏。可以使用性能警报来达到这个目的。

性能警报基于性能计数器，这些计数器与 Windows 性能监视器程序中的性能计数器相同。我们提供与 SQL Server 的各种构件有关的统计信息，并对那些构件进行操作。处理填满的事务泵是使用性能警报的一个恰当例子。因为当事务日志填充到 100%时，任何用户都将无法访问数据，因此而无法工作。出现了这种情况将无疑会给公司造成损失，因此，应当事先发现问题，在事务日志达到一定比例或规定的上限时立即将其清除。

为了说明性能警报的作用，我们来创建一个性能警报，该警报在【db_stu】数据库的日志满时被激活。在用户自己的系统上，应当将这个警报设置成当事务日志达到 80%时激活一个备份事务日志的作业。具体创建警报的步骤如下。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接服务器。

（2）在【对象资源管理器】窗口中，展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【警报】节点，从弹出的菜单中选择【新建警报】命令，打开【新建警报】窗口。

（4）在【名称】文本框中输入 Performance Alert；在【类型】下拉列表中选择【SQLS 性能条件警报】选项；在【对象】下拉列表中选择【SQL Server Database】选项；在【计数器列表中选择【Percent Log Used】选项；在【实例】下拉列表中选择【db_stu】选项；将【计数器满足以下条件时触发警报】设置为【低于】，在【值】文本框中输入 100。

（5）选择【响应】选项，打开【响应】选项窗口，选中【通知操作员】复选框，并选中 Administrator 行的【Net Send】复选框。

（6）单击【确定】按钮警报创建完成。

（7）当 Net Send 消息弹出时，仔细观察它给出的细节，包括错误号、描述和其他内容，然后单击【确定】按钮关闭消息。

10.6.4　WMI 警报

WMI（Windows Management Instrumentation）是 Microsoft 对基于企业管理 WebBased Enterprise Management 标准的实现。这是一个行业新标准，通过将系统、应用程序和网络之类的管理构件显示为一组常用对象而使系统变得更容易管理。SQL Server 已经更新成使用 WMI 并响应 WMI 事件。

利用 WMI 警报，可以响应以前可能从未见过的事件。例如，可以将警报创建成在运行 CREATE TABLE 语句时激活，这样就可以跟踪数据库上的存储情况了；或者在发布 ALTER LOGIN 命令时激活，这样对管理安全性是非常有用的。所以有效利用 WMI 警报是非常有好处的。下面，我们就介绍一下如何创建 WMI 警报。

（1）在【SQL Server Manager Studio】的【资源管理器】中，展开【服务器】及其下面的【SQL Server 代理】节点。

（2）右击【警报】节点，在弹出的菜单中选择【新建警报】命令，打开【新建警报】窗口。

（3）在【名称】文本框中输入 WMI Alert；在【类型】下拉列表中选择【WMI 事件警报】选项；确保【命名空间】文本框是默认值。

（4）在【查询】文本框中输入如下语句。

```
SELECT * FROM DDL_DATABASE_LEVEL_EVENT
WHERE DatabaseName='adventureWorks'
```

（5）选择【响应】选项，打开【响应】选项窗口，选中【通知操作员】复选框，并选中 Administrator 行的【Net Send】复选框。

（6）选择【选项】选项，打开【选项】选项窗口，选中【警报错误文本发送方式】下面的【Net Send】复选框。

（7）在【SQL Server Management Studio】中，单击【新建查询】按钮，打开一个新的 SQL Server 查询窗口。

（8）输入并执行下面的语句来激活这个警报：

```
USE　AdventureWorks
ALTER TABLE　Person.Address　ADD WMI_Test_Column VARCHAR(20) NULL
```

（9）当 Net Send 消息弹出时，仔细观察它给出的细节，包括错误号、描述和其他内容，单击【确定】按钮关闭消息。

（10）如果想让 AdventureWorks 数据库恢复到正常状态，可以执行下面的语句。

```
USE AdventureWorks
ALTER TABLE    Person.Address DROP WMI_Test_Column
```

最后，当已经创建的警报不再需要时，可通过下面的方法来禁用警报：

（1）在【SQL Server Management studio】的【资源管理器】中，展开【服务器】及其下面的【SQL Server 代理】节点。

（2）展开【警报】节点，右击打算禁用的警报名称，从弹出的菜单中选择【属性】命令，打开【警报属性】窗口。

（3）禁用【启用】复选框，单击【确定】按钮完成修改操作。

10.7　作　　业

作业是一系列由 SQL Server 代理按顺序执行的指定操作。一个作业可以执行各种类型的活动，包括运行 Transact-SQL 脚本、命令提示符应用程序、Microsoft ActiveX 脚本、Integrations 包、Analysis Services 命令和查询或复制任务等。作业可以运行重复或可计划的任务，然后通过生成警报来自动通知用户作业状态，从而极大地简化了 SQL Server 的管理。

10.7.1　概述

作业就是一个任务系列，其中的任务能被自动化为在需要它们的任何时候运行。SQL Server 上的任何一个作业也同样如此。例如，可将创建数据库分为以下几个步骤。第一步创建数据库。第二步备份新的数据库，因为备份对数据库（特别是大型数据库）是非常必要的。第三步在完成数据库的备份之后，可以在数据库中创建一些表，最后再向表中导入相关的数据。这几个步骤，每一步都是一项任务，这些任务之间都是独立的，并且前一步没有完成，后一步就不能开始。但是，并不是所有的任务都是这样。

通过控制各个步骤的流程，可以将纠错机制内建到作业中来。例如，在上述的创建数据库的作业中，每一步都有一个简单的逻辑，用于规定“成功时转入下一步，失败时退出作业”。如果在创建的过程中硬盘被最终填满，则作业停止。如果在作业的末尾创建一个用于清理硬盘空间的步骤，就可以创建这样一个简单的逻辑，用于规定“第一步失败，转到第五步；如果第五步成功，则返回到第一步”。有了这些步骤之后，就可以通知 SQL Server 何时启动这个作业了。

要通知 SQL Server 何时运行作业，需要创建执行计划，而且这方面有很大的灵活性。如如果一个作业只是创建一个数据库，运行这个作业多次没有太大意义，因此只需创建一个在数小时后激活该作业的执行计划即可。如果正在创建一个用来执行事务日志备份的作业，则需要一个不同的执行计划。

作业不仅可以计划成在一天的某些时间激活，还可以计划成在一星期或一个月的某些天激活。作业可以计划成每当 SQL Server 代理服务启动时就运行一次，甚至可以计划成每当处理器空闲时就运行一次。

执行计划可以设置成在一定的时间量之后终止。因此，如果知道一个作业在几周后终止，则可以设置它的终止日期，那么到时它将自动终止计划。在创建作业时，可以给作业添加在成功、失败或完成时接收通知的操作员（不管作业成功还是失败）。那么当作业结束时，操作员就可以

收到来自作业的输出结果，这在运行的作业对服务器或应用程序至关重要时是非常有用的。

由于能够改变步骤的逻辑流程、安排作业在需要时运行，以及让作业在完成时发出通知，因此作业可能会很复杂。所以为了更好地创建作业，可以在创建作业之前先进行一下规划，这样再去创建它们就变得容易多了。

SQL Server 使用了两种类型的作业：本地和多服务器作业。下面将分别对这两种类型的作业进行介绍。

10.7.2　创建本地作业

本地作业是包含一系列步骤和执行计划的标准。这些作业只能运行在创建它们的计算机上，因此称为本地作业。下面，将创建一个本地作业，该作业用于创建一个新的数据库，然后备份该数据库。具体过程如下。

（1）从【开始】菜单上选择【程序】→【Microsoft SQL Server 2005】→【SQL Server Management Studio】 命令，打开【SQL Server Management Studio】窗口，并使用 Windows 或 SQL Server 身份验证建立连接。

（2）在【对象资源管理器】窗口中展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【作业】节点，从弹出的菜单中选择【新建作业】命令，打开【新建作业】窗口。

（4）在【名称】文本框中输入 Create and Backup Database Job；【所有者】文本框保持默认值；从【类别】下拉列表中选择【未分类（本地）】选项。

（5）选择【步骤】选项，打开【步骤】选项窗口，单击【新建】按钮打开【新建作业步骤】窗口。

（6）在【步骤名称】文本框中输入 Create Database；从【类型】下拉列表中选择【Transact-SQL 脚本（T-SQL）】选项，然后在【命令】文本框中输入如下语句，用于在 C 盘驱动器 SQLDB 文件夹下创建一个名为 Test 的数据库：

```
CREATE  DATABASE  Test
ON PRIMARY(NAME=Test_data，
FILENAME='C:\sqldb\Test.mdf'，
SIZE=10MB，
MAXSIZE=15，
FILEGROWTH=10%)
```

（7）单击【分析】按钮验证语句输入是否正确，然后选择【高级】选项，打开【高级】选项窗口。

（8）确保从【成功时要执行的操作】下拉列表中选择【转到下一步】选项，以及从【失败时要执行的操作】下拉列表中选择【退出报告失败的作业】选项，其他设置保持默认值。单击【确定】按钮返回。

（9）创建作业的第二步，单击【新建】按钮，再次打开一个【新建作业步骤】窗口。

（10）在【步骤名称】文本框中输入 Backup Database；从【类型】下拉列表中选择【Transact-SQL 脚本（T-SQL）】选项，输入语句来备份新创建的 Test 数据库。

（11）设置完成后单击【确定】按钮返回。

（12）选择【计划】选项，打开【计划】选项窗口。单击【新建】按钮打开【新建作业计划】窗口，用来创建一个执行计划来通知 SQL Server 执行该作业。

（13）在【名称】文本框中输入 Create and Backup Database；在【计划类型】下拉列表中选择【执行一次】选项，然后设置想让作业开始执行的时间和日期，这里设定为当前系统时间加上 10 分钟。

（14）设置完成后单击【确定】按钮返回。

（15）选择【通知】选项，打开【通知】选项窗口。因为前面已经配置了“数据库邮件”，所以选中【电子邮件】和【Net Send】复选框，并选择 Administrator 作为接到通知的操作员，选择【当作业完成时】选项作为通知的条件（当作业完成时包括了当作业成功时和当作业失败时）。

（16）单击【确定】按钮创建作业。此时，作业就会在上面设定的时间开始执行了，具体情况可以通过右击某一作业，从弹出的菜单中选择【查看历史记录】命令，打开【日志文件查看器】进行验证。

上面创建一个作业包含了两个步骤，第一步创建了一个名为 Test 的新数据库，第二步备份了新创建的 Test 数据库。该作业只运行一次，最后不管成功与否都将通知 Administrator。这个作业的两个步骤都使用了 Transact-SQL 语句来实现，其实还可以使用其他的脚本语句，如 JavaScript、VBScript 等。这样就可以将一些用 Transact-SQL 语句无法实现的作业用其他的脚本语言来实现。

下面来创建一个打印一条语句的作业。

（1）从【开始】菜单上选择【程序】→【Microsoft SQL Server 2005】→【SQL Server Management Studio】命令，打开【SQL Server Management Studio】窗口，并使用 Windows 或 SQL Server 身份验证建立连接。

（2）在【对象资源管理器】窗口中展开【服务器】节点，然后展开【SQL Server 代理】节点。

（3）右击【作业】节点，从弹出的菜单中选择【新建作业】命令，打开【新建作业】窗口。

（4）在【名称】文本框中输入 VBScript Test，其他项保持默认值。

（5）选择【步骤】选项，打开【步骤】选项窗口，单击【新建】按钮，打开【新建作业步骤】窗口。

（6）在【步骤名称】文本框中输入 Print；从【类型】下拉列表中选择【ActiveX 脚本】选项；在【命令】文本框中输入 VBScript 脚本语句。

（7）单击【确定】按钮完成作业步骤的创建。

（8）选择【计划】选项，打开【计划】选项窗口，单击【新建】按钮，打开【新建作业计划】窗口。

（9）在【名称】文本框中输入 Print Time；从【计划类型】下拉菜单中选择【执行一次】选项；然后设置作业执行的时间和日期。

（10）单击【确定】按钮完成新建作业计划。

（11）选择【通知】选项，打开【通知】选项窗口，设置完成【作业完成时要执行的操作】选项后，单击【确定】按钮创建作业。

（12）通过【日志文件查看器】窗口查看作业的执行情况。

其实每个作业的历史信息都存放在数据库中，默认情况下，总共可以存放 1 000 条历史记

录，每个作业最多占用 100 条记录。根据需要可以改变这些默认值，具体步骤如下。

（1）从【开始】菜单上选择【程序】→【Microsoft SQL Server 2005】→【SQL Server Management Studio】命令，打开【SQL Server Management Studio】窗口，并使用 Windows 或 SQL Server 身份验证连接服务器。

（2）在【对象资源管理器】窗口中展开【服务器】节点，右击【SQL Server 代理】节点，在弹出的菜单中选择【属性】命令，打开【SQL Server 代理属性】窗口，然后选择【历史记录】选项，打开【历史记录】选项窗口。

（3）根据需要重新设置【当前作业历史记录日志的大小】选项区域的内容，最后单击【确定】按钮完成修改操作。

10.7.3　创建多服务器作业

在对单个服务进行管理时，创建本地服务管理是非常有好处的，但现在越来越多的企业拥有多个数据库服务器。这些服务器每个可能都需要作业，并且有些作业是服务器独有的，有些作业是重复的，如果在每个服务器上分别创建本地作业，这个过程将既费时又不容易管理。所以我们使用另一种更好的办法就是创建多服务器作业。

多服务器作业只需在一个服务器上创建一次，然后通过网络下载到运行该作业的其他服务器上即可。在创建多服务器作业之前，必须先指定两种类型的服务器：主服务器和目标服务器。主服务器（简称 MSX）是创建和管理多服务器作业的地方，目标服务器每隔一定时间向主服务器轮询一次作业，并下载这些作业，然后按预定的时间运行它们。

1．指定服务器

下面详细介绍如何创建、指定主服务器和目标服务器。

（1）打开【SQL Server Management Studio】，并使用 Windows 或 SQL Server 身份验证连接到服务器。

（2）在【对象资源管理器】窗口中，展开【服务器】节点，右击【SQL Server 代理】节点，从弹出的菜单中选择【多服务器管理】→【将其设置为主服务器】命令，打开【主服务器向导】窗口。

（3）单击【下一步】按钮，打开【主服务器向导】的【主服务器操作员】窗口。在【电子邮件】文本框中输入前面已经配置好的操作员电子邮件地址 xieke@cybertang.com，在【Net Send】文本框中输入计算机名称 computer。

（4）单击【下一步】按钮，打开【主服务器向导】的【目标服务器】窗口，从【已注册服务器】列表中选择一个实例名称，将其添加到【目标服务器】列表中，它将接受来自主服务器的作业。

（5）单击【下一步】按钮，打开【检查服务器兼容性】窗口，若测试结果有错误，将无法继续下一步操作，需要纠正错误；反之，则单击【关闭】按钮。

（6）在【主服务器向导】的【主服务器登录凭据】窗口中，选中【在必要时创建新登录名，并为其分配针对 MSX 的权限】复选项。

（7）单击【下一步】按钮，打开【主服务器向导】的【完成该向导】窗口。

（8）单击【确定】按钮完成创建主服务器和指定目标服务器。

2．创建多服务器作业举例

在上面已经创建了一个主服务并指定了一个目标服务器，下面就在主服务器中创建一个作

业，该作业将在目标服务器上运行并在运行结束时通知主服务器操作员。

（1）在【资源管理器】窗口展开【SQL Server 代理】节点，右击【作业】节点，从弹出的菜单中选择【新建作业】命令，打开【新建作业】窗口。

（2）在【名称】文本框中输入 Servers Job Test，其他选项保持默认值。

（3）选择【步骤】选项，打开【步骤】选项窗口，单击【新建】按钮，打开【新建作业步骤】窗口。

（4）在【步骤名称】文本框中输入 Create Database；从【类型】下拉列表中选择【Transact-SQL 脚本（T-SQL）】选项；在【命令】文本框中输入 T-SQL 语句，用于在 C 盘驱动器 SQLDB 文件夹下创建一个名为 Servers_Test 的数据库。

（5）单击【确定】按钮创建该步骤。

（6）选择【计划】选项，打开【计划】选项窗口，单击【新建】按钮，打开【新建作业计划】窗口。

（7）在【名称】文本框中输入 Create Database；从【计划类型】下拉列表中选择【执行一次】选项，并设定开始执行的日期和时间，单击【确定】按钮创建该计划。

（8）选择【通知】选项，打开【通知】选项窗口。选中【Net Send】复选框，并选择 Administrator 作为接到通知的操作员，选择【当作业成功时】选项作为通知的条件。

（9）选择【目标】选项，打开【目标】选项窗口，选中【目标为本地服务器】单选按钮。最后单击【确定】按钮创建这个作业。

此时，在主服务器上创建了一个作业，然后它被下载到目标服务器上运行，并创建了一个数据库。但是，这个作业是如何到达目标服务器的呢？其实目标服务器已默认地配置为每隔 60 秒向主服务器轮询作业一次，也就是当主服务器有作业时就下载下来，按作业的设定时间运行。当然，可以通过 sp_post_msx_operation 存储过程调整轮询的间隔以符合工作的要求。如果想同步一个名为 Target 的目标服务器上的时钟可以使用下列语句：

```
sp_post_msx_operation='SYNC-TIME'，@object_type='server'，
@specific_target_server='Target'
```

如果将 Target 服务器上的轮询间隔改为 100 秒，可以使用下列语句：

```
sp_msx_operation='SET—POLL'，@object_type='server'，
@specific_target_server='target'，@value='100'
```

上面已经了解了操作员、警报和作业的概念，并掌握了如何创建操作员、警报和作业的方法，现在已经可以使用它们来自动化管理数据库了。下面介绍如何使用“维护计划向导”。

10.8　维护计划向导

为了使数据库一直保持最佳的运行状态需要执行许多的任务，如数据库与事务日志备份、优化数据库等，这些必须定期执行，才能使服务器保持正常运行。我们可以通过创建作业来执行它们，但是必须为每个数据库都创建许多的作业才行，这是相当麻烦和辛苦的。其实我们完全不必这样，只需通过使用“维护计划向导”即可达到使数据库保持正常运行的目的。

“维护计划向导”可以用来为需要在数据库上定期执行的所有标准维护任务创建作业。下面介绍使用“维护计划向导”的过程。

（1）从【开始】菜单上选择【程序】→【Microsoft SQL Server 2005】→【SQL Server Management Studio】命令，打开【SQL Server Management Studio】窗口，并使用 Windows 或 SQL Server 身份验证建立连接。

（2）在【对象资源管理器】窗口中展开【服务器】节点，然后展开【管理】节点。

（3）右击【维护计划】节点，从弹出的菜单中选择【维护计划向导】命令，打开【维护计划向导】初始窗口。

（4）单击【下一步】按钮，打开【维护计划向导】的【选择目标服务器】窗口。

（5）在【名称】文本框中输入 MaintenancePlan 1；选择【服务器】为【LJH】；这里选中【使用 Windows 身份验证】单选按钮，单击【下一步】按钮，打开【维护计划向导】的【选择维护任务】窗口。

（6）这里选择【选择一项或多项维护任务】列表中的所有维护任务。单击【下一步】按钮，打开【维护计划向导】的【选择维护任务顺序】窗口。

（7）可以根据需要调整维护项目的顺序，这里保持默认顺序。单击【下一步】按钮，打开【维护计划向导】的【定义“数据库检查完整性”任务】窗口。

（8）单击【数据库】下拉列表框，选中【以下数据库】单选按钮，选中【db_stu】数据库复选框。其中，【所有数据库】表示该选项将服务器上所有数据库都包含在同一个维护计划中；【所有系统数据库】表示该选项包含除了系统之外的所有数据库；【以下数据库】选项允许选择要包含在维护计划中的数据库。

（9）选择【db_stu】数据库选项后单击【确定】按钮返回。

（10）单击【下一步】按钮，打开【维护计划向导】的【定义“收缩数据库”任务】窗口。同步骤（8）一样设定【数据库】选项。该窗口用于指定在数据库变得太大时应该如何缩小数据库，以及何时开始缩小、缩小多少和收缩后的可用空间如何使用等。

（11）单击【下一步】按钮，打开【维护计划向导】的【定义“重新组织索引”任务】窗口。同步骤（8）一样设定【数据库】选项；从【对象】下拉列表中选择【表】选项，设置【选择】选项为【<选择-项或多项>】。

（12）设定好以后，单击【确定】按钮返回【维护计划向导】的【定义“重新组织索引”任务】窗口。单击【下一步】按钮，打开【维护计划向导】的【定义“重新生成索引”任务】窗口。同前面所述的方法一样对其中选项进行设置。其中，【使用默认可用空间重新组织页】表示该选项填充因子重新产生页面；【将每页的可用空间百分比改为】表示为该选项创建一个新的填充因子，如果将它设置为 20，则页面将包含 20%的自由空间。

（13）单击【下一步】按钮，打开【维护计划向导】的【定义“更新统计信息”任务】窗口。统计信息基于一个值在列中出现的次数，而且由于列中的值是会变化的，所以统计信息需要更新以反映那些变化。同样，从【数据库】下拉列表中选择【db_stu】数据库，从【对象】下拉列表中选择【表和视图】选项，下面的选项保持默认值。

（14）单击【下一步】按钮，打开【维护计划向导】的【定义“清除历史记录”任务】窗口。该窗口用于设置何时和如何清理数据库的历史记录，以使它保持正常运行。根据实际需要设置，这里保持默认设置。

（15）单击【下一步】按钮，打开【维护计划向导】的【定义“执行 SQL Server 代理作业”任务】窗口，选择 Create and Backup Database Job 作业。

（16）单击【下一步】按钮，打开【维护计划向导】的【定义“备份数据库”任务】窗口，

从【数据库】下拉列表中选择要备份的数据库，设定其他备份选项及备份数据库的路径。该窗口用于指定哪些数据库将得到备份、备份的方式，以及它们将被备份到什么地方。

（17）单击【下一步】按钮，接下来的两个窗口分别是【维护计划向导】的【差异备份数据库】窗口和【事务日志备份】窗口。在这两个窗口中用与步骤（16）相同的方法进行设置。

（18）单击【下一步】按钮，打开【维护计划向导】的【选择计划属性】窗口，更改【计划】选项。

（19）单击【下一步】按钮，打开【维护计划向导】的【选择报告选项】窗口。这里可以设定作业在每次运行时将一个报告写到文本文件上，并且可以通过电子邮件将报表发送给操作员。在本例中，选中【将报告写入文本文件】复选框，保持文件夹位置为默认；然后选中【以电子邮件形式发送报告】复选框，从【收件人】下拉列表中选择【Administrator】选项。

（20）单击【下一步】按钮，打开【维护计划向导】的【完成该向导】窗口。该窗口中显示了待执行的任务的汇总信息。

（21）单击【完成】按钮创建这个维护计划。

（22）当 SQL Server 创建完成维护计划后，单击【关闭】按钮即可完成维护计划向导。

如果在创建了维护计划之后的某个时候需要修改它，可以在【SQL Server Management Studio】的【资源管理器】窗口中展开【管理】节点下面的【维护计划】节点，右击【MaintenancePlan 1】维护计划，并从弹出的菜单中选择【修改】命令，打开【MaintenancePlan 1 设计】窗口。在该窗口中可以修改维护计划。

如果要查看 MaintenancePlan 1 维护计划的历史记录，可以右击【MaintenancePlan 1】维护计划，从弹出的菜单中选择【查看历史记录】命令，便可以显示维护计划最后执行的一切任务。

可以看出，维护计划在保证数据库顺利和有效运行方面是很有帮助的。需要备份数据库和日志、重新组织数据库文件内的索引、定期检查数据库的完整性等都可以使用“维护计划向导”来完成。

习　　题

1．复制有哪几种类型，它们的特点是什么？

2．哪些情况下事务会被终止？

3．为什么要创建操作员？操作员在 SQL Server 自动化管理中起什么作用？

4．使用“数据库邮件”有哪些优点？

5．使用“维护计划向导”能够执行哪些维护任务？

第 11 章　VB/SQL Server 开发与编程

本章将简单介绍利用 Visual Basic 开发数据库应用程序的各种技巧。例如，介绍 Visual Basic 访问数据库的方法，配置 ODBC 数据源的详细步骤，可视化数据管理器的使用，最后以一个学生信息管理应用程序为例，介绍使用 Visual Basic 开发 SQL Server 数据库应用程序的思路和方法。

11.1　Visual Basic 数据库访问方法

Visual Basic 环境提供的数据库访问方法包括了 Data 控件、DAO、ADO、RDO、ADC、RDC、JET、OLEDB 和 VBSQL 等。本节将有选择地讨论数据库访问方法，重点以 Data 控件和 ADO 访问方法为例，详细讨论 Data 数据控件和 ADO 的关键对象。

11.1.1　Data 控件访问 SQL Server 数据库

数据控件 Data 是 Visual Basic 开发数据库时经常使用到的控件对象，它主要用来打开并且访问数据库，Data 控件提供的访问数据库的方法，可以不编写代码，只需要设置它的 Connect 属性及 Record Source 属性便可以对 Visual Basic 所支持的数据库执行基本的数据访问操作。由于 Data 控件是由 Microsoft 的 Jet 数据引擎来实现数据库的访问的，这使用户可以方便地访问许多 Visual Basic 所支持的数据库格式。

数据控件的使用可以加快 Visual Basic 数据库应用程序的开发，只需要编写很少的代码，就可以实现一个数据库应用程序，并可以把一些数据控件、DataGrid 和命令按钮放在一个窗体中，当设置好它们的一些简单的属性后，就可以方便地访问数据库了。如果要实现更多的功能，就需要编写更多的代码，然后把这些不同功能的代码加到不同的 Data 控件的不同事件中，就可以建立一个具有强大功能的数据库应用程序了。

要使用这些专业的 Data 控件方便快速地访问数据库，还需要进行数据绑定。DataSource 和 DataField 这两个属性就决定了绑定发生时控件的一些行为特征。其中 DataSource 属性指定了控件将要绑定到数据库的名字，DataField 属性指定了将要绑定到的数据库中表的字段名。

11.1.2　ADO 访问 SQL Server 数据库

Visual Basic 与 SQL Server 连接的方式多种多样，从数据连接的机制讲，有 DAO、ADO、RDO 等，从连接的层次上讲，有两层连接、三层连接和多层连接等。目前，在一些大型的数据库开发系统中，ADO 是一种应用比较广泛的方法。它集成了 DAO 和 RDO 的优点，也支持多层结构的开发模式，可以通过简单的编程实现。

什么是 ADO？ADO 就是 ActiveX Data Object 的缩写，意思就是动态数据对象，利用它可以将应用程序方便地动态连接到数据库中。

ADO 的关键对象包括以下 3 项。

- Connection 代表了实际的数据库的连接；
- Command 代表在数据库连接中执行的查询；

● Recordset 代表通过执行 Command 对象查询而获取的记录的集合。

1．连接对象——Connection

Connection 对象代表了一个到指定的数据源的成功连接。应用程序通过连接访问数据源，如果连接已经成功，Connection 就会以对象的形式存在。程序中在建立连接的时候，还需要设置连接的字符串，包括了指定连接数据库的驱动程序，连接到的数据库的名称，用户名称和密码等。

● ConnectionString 属性：连接字符串，在打开连接前需要进行的设置。
● CursorLocation 属性：设置或者返回指针的位置。
● DefaultDatabase 属性：为连接指定一个默认的数据库。
● Isolationlevel 属性：连接的间隔等级。
● Open 方法：打开一个连接。
● Close 方法：关闭一个连接。
● Execute 方法：在连接上执行命令。

2．命令对象——Command

Command 对象是数据源要执行的一系列 SQL 操作的定义。使用 Command 对象可查询数据库并返回数据记录集对象的查询结果。Command 对象可以完成对数据的各种操作，包括了 SELECT、UPDATE 和 DELETE 等。

● CommandText 属性：定义 Command 对象的 SQL 语句。
● CommandType 属性：优化命令效率。
● CommandTimeout 属性：服务器响应命令的时间。
● Execute 方法：执行命令并返回一个数据记录集。

3．数据集对象——Recordset

数据集对象定义了从数据库返回的一个记录集合，结果以表的形式组织。通过数据记录集可以对记录进行各种操作。

● RecordCount 属性：返回记录集中的条数。
● BOF、EOF 属性：记录集中游标的当前位置是否是记录集头或者记录集尾。
● MoveNext 方法：将记录集中游标向后移动一个位置。
● MovePre 方法：将记录集中游标向前移动一个位置。
● MoveFirst 方法：将记录集中游标移动到最前一个位置。
● MoveLast 方法：将记录集中游标移动到最后一个位置。

ADO 的使用步骤简单又符合逻辑。ADO 可以通过下面的几个步骤来完成对数据库的访问操作。

● 创建一个数据库连接对象 Connection。
● 创建一个命令行对象 Command。
● 执行 SQL 语句。
● Select 语句返回的数据保存在数据集对象 Recordset 中，便于其他 SQL 语句操作。
● 通过对 Recordset 数据集对象进行各种操作更新数据源，如插入、修改和删除。
● 结束事务关闭连接。

11.2　ODBC 数据源配置和可视化数据管理器

ODBC 即 Open Database Connectivity，它是处理各种数据库的通用的 API 函数的产物，在程序中通过它可以使用相同的代码访问不同的数据库。ODBC 的基本思想就是为用户提供简单、标准的数据库连接的公共编程接口。

Visual Basic 提供了一个功能强大的数据库管理工具——可视化数据管理器。利用该管理器可以创建数据库、维护数据库、修改数据、建立数据查询、动态构造 SQL 查询语句等。

11.2.1　ODBC 数据源配置

使用 Windows 控制面板中的 ODBC 数据库管理工具，可以为 SQL Server 数据库创建一个 myvbsql 的数据源。操作步骤如下。

（1）选择【开始】→【设置】→【控制面板】→【管理工具】→【数据源 ODBC】命令，打开【ODBC 数据源管理器】对话框，如图 11-1 所示。

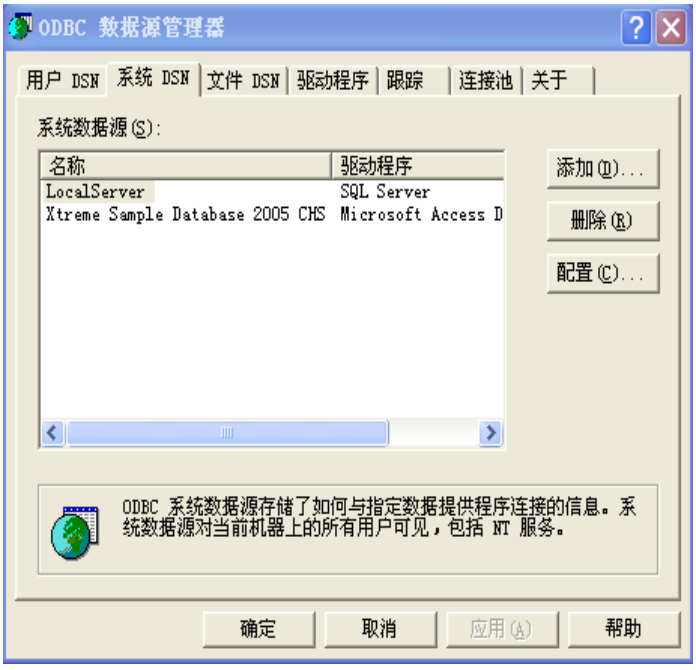


图 11-1　OBDC 数据源管理器

（2）单击【添加】按钮，进入【创建新数据源】对话框，如图 11-2 所示，选择【SQL Server】数据源驱动程序，单击【完成】按钮，进入到“创建到 SQL Server 的新数据源”对话框。

（3）在“创建到 SQL Server 的新数据源”向导的第一个对话框中，输入名称“myvbsql”和所要连接的 SQL Server 名称，即完整的计算机名称，如图 11-3 所示。

图 11-2　创建新数据源

图 11-3　数据源名称

（4）单击【下一步】按钮进入向导的第二个对话框，如图 11-4 所示。选择以 SQL Server 验证的方式登录 SQL Server，输入 SQL Server 数据库管理系统的默认用户名称为 sa，密码可以设置为空。

（5）单击【下一步】按钮，进入向导的第三个对话框，如图 11-5 所示。更改默认的数据库为我们建立的【db_stu】。

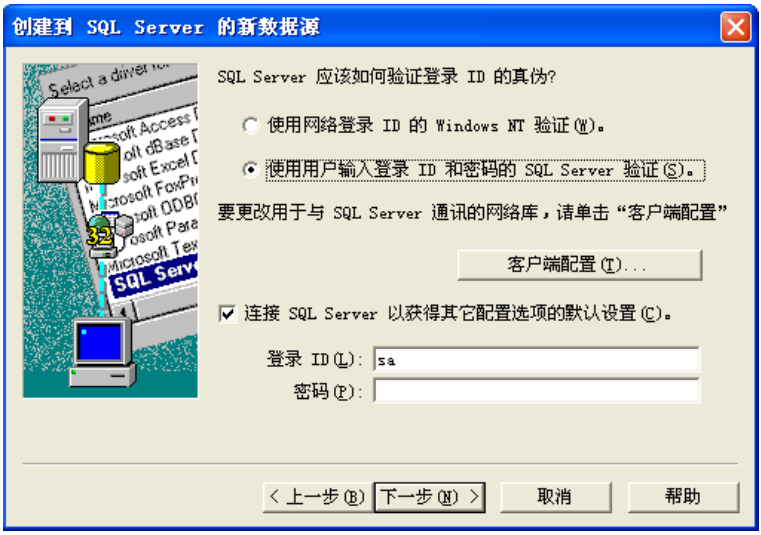


图 11-4　SQL Server 登录方式

图 11-5　更改默认数据库

（6）单击【下一步】按钮，进入向导的第四个对话框，如图 11-6 所示。单击【完成】按钮，进入向导的最后一个对话框，如图 11-7 所示。测试数据源连接情况，连接成功单击【确定】按钮，确定 myvbsql 数据源配置，如图 11-8 所示。返回 ODBC 数据源管理器，这个时候可以发现 myvbsql 已经出现在了 ODBC 数据源管理器的列表中，如图 11-9 所示。这个时候 SQL Server 数据源 myvbsql 就创建完成了。

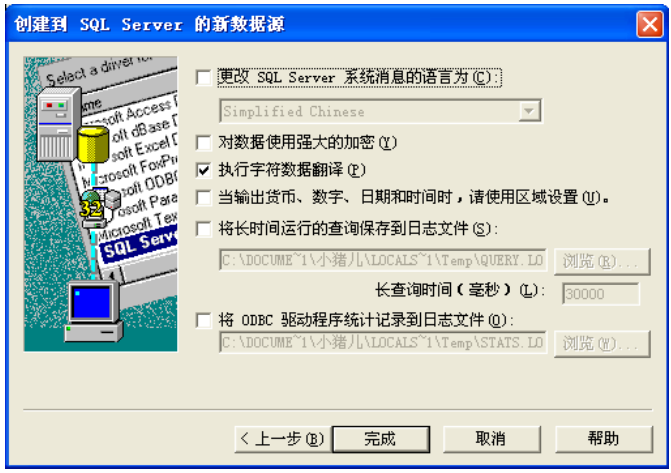


图 11-6　选择附加信息

图 11-7　数据源配置信息

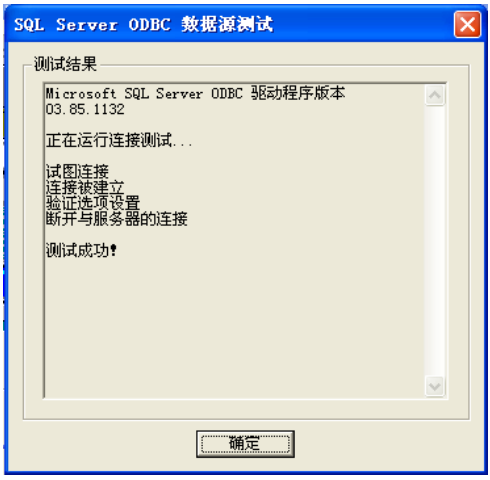


图 11-8　数据源连接测试成功

图 11-9　创建好的数据源

11.2.2　可视化数据管理器

Visual Basic 提供的可视化数据管理器，可以实现对数据库的管理工作，包括了创建数据表、维护数据库结构、修改数据、建立查询等操作。在 Visual Basic 主菜单【外接程序】中选择【可视化数据管理器】，就可以进入如图 11-10 所示的可视化数据管理器界面，进行数据库的管理操作了。

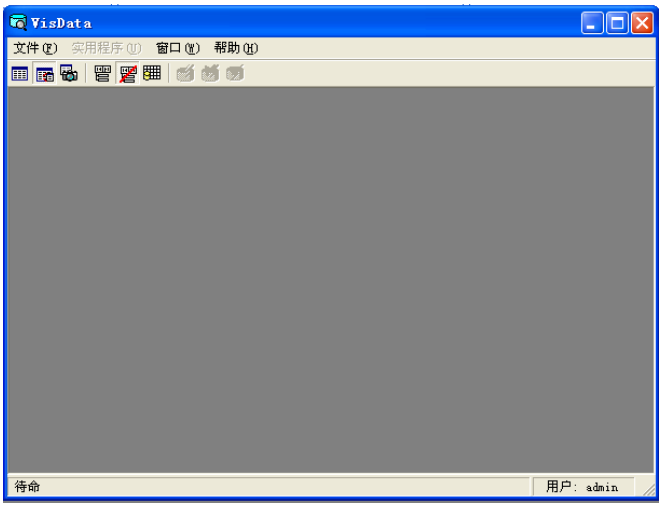


图 11-10　可视化数据管理器

（1）打开 ODBC 数据库，在【可视化数据管理器】界面中选择【文件】菜单中的【打开数据库】命令，再选择【ODBC】命令，打开【ODBC 登录】对话框，如图 11-11 所示。

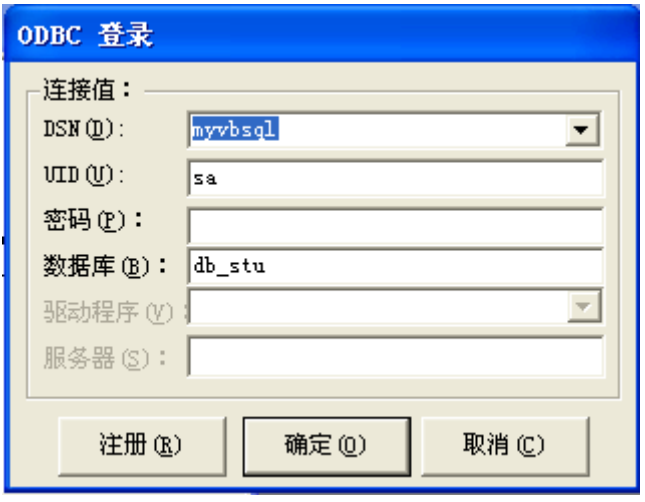


图 11-11　【ODBC 登录】对话框

（2）在弹出的【ODBC 登录】对话框中，选择连接的 ODBC 数据源【myvbsql】，用户名称【sa】，密码为空，数据库名称为【db_stu】，如图 11-11 所示。单击【确定】按钮即可打开【myvbsql.db_stu】数据库界面，如图 11-12 所示。

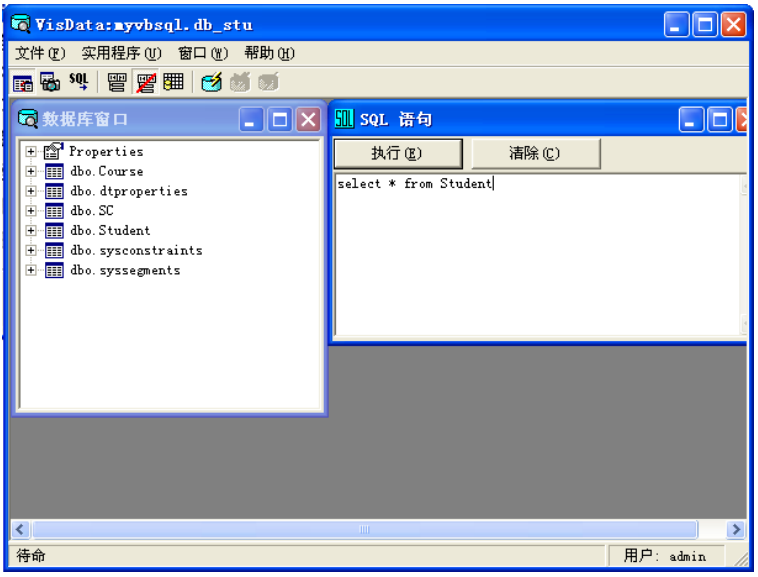


图 11-12　myvbsql.db_stu 数据库界面

（3）在“SQL 语句”中输入要查询、修改或者删除的 SQL 语句，就可以轻松实现对相应的数据库表的操作。例如，在“SQL 语句”中输入“select * from Student”就可以实现对数据库【db_stu】中【Student】表的查询操作，查询的结果如图 11-13 所示。

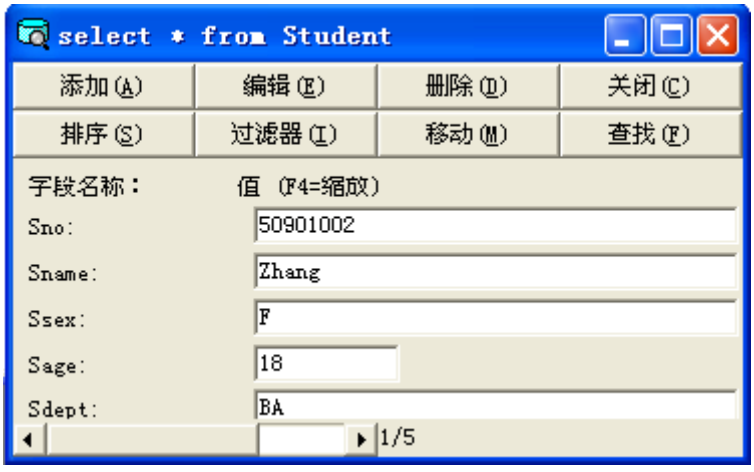


图 11-13　查询结果

（4）选择【可视化数据库管理器】的【文件】菜单中的【新建】命令，也可以实现新建一个数据库；选择【添加字段】命令可以实现对数据库中的表字段的建立，由于和 SQL Server 中数据库的建立和表的建立类似，这里就不再一一叙述了。

11.3　VB/SQL Server 编程——学生信息管理系统

本节我们以学生信息管理系统为例，采用 Visual Basic 和 SQL Server 相结合的方式，利用 Visual Basic 中的 Data 数据控件和 ADO 技术实现对 SQL Server 数据库的访问、查询、修改、

删除和搜索等操作。

在 SQL Server 上已经建立好了学生信息数据库，名称为【db_stu】。它包括了 3 个数据表，学生表【Student】、课程表【Course】，以及选课表【SC】，3 个数据表的字段添加和前面章节相同，这里不对创建过程进行详述。

11.3.1　用户界面设计

（1）创建主界面。首先打开 Visual Basic，新建一个【标准 EXE】工程，然后设计学生信息管理主界面，如图 11-14 所示。在 Visual Basic 中选择【工具】，再选择【菜单编辑器】命令，对 Form1 的菜单栏进行设置，添加如图 11-15 所示的菜单名。主界面各控件的属性设置和修改如表 11-1 所示。

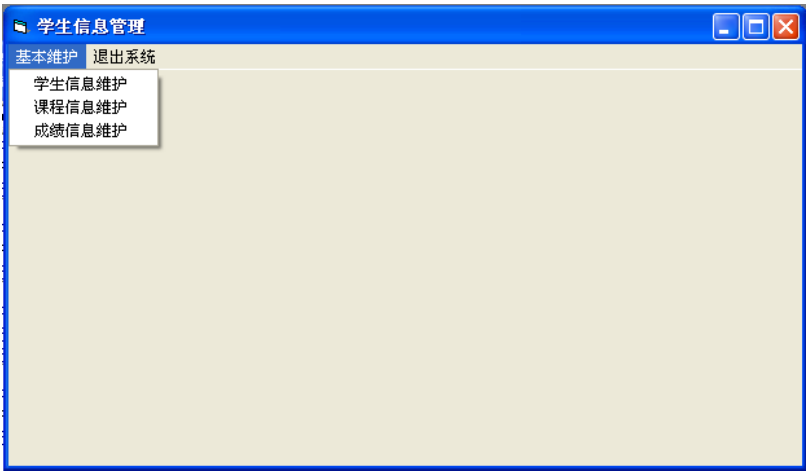


图 11-14　学生信息管理主界面

图 11-15　菜单编辑器

表 11-1　主界面控件属性设置

控 件 名 称	属　　性	修　改　值	功　　能
Form1	Caption	学生信息管理	主界面
菜单编辑器	标题	基本维护	一级菜单
	名称	menu1	信息维护主菜单
	标题	学生信息维护	二级菜单
	名称	S_menu	打开学生信息维护界面
	标题	课程信息维护	二级菜单
	名称	C_menu	打开课程信息维护界面
	标题	成绩信息维护	二级菜单
	名称	G_menu	打开成绩信息维护界面
	标题	退出系统	一级菜单
	名称	menu2	退出系统

（2）创建子界面。由于 3 个表的操作界面类似，所以这里我们只以【Stuednt】表的访问和操作为例进行详细叙述，其他两个表，读者进行简单的修改就可以实现。选择【工程 1】单击鼠标右键，选择【添加】，再选择【添加窗体】命令，选择【窗体】，单击【打开】按钮，这样就可以为工程添加一个新的窗体，这个窗体就可以作为学生信息维护的子窗体。为窗体添加控件布局如图 11-16 所示，属性修改如表 11-2 所示。

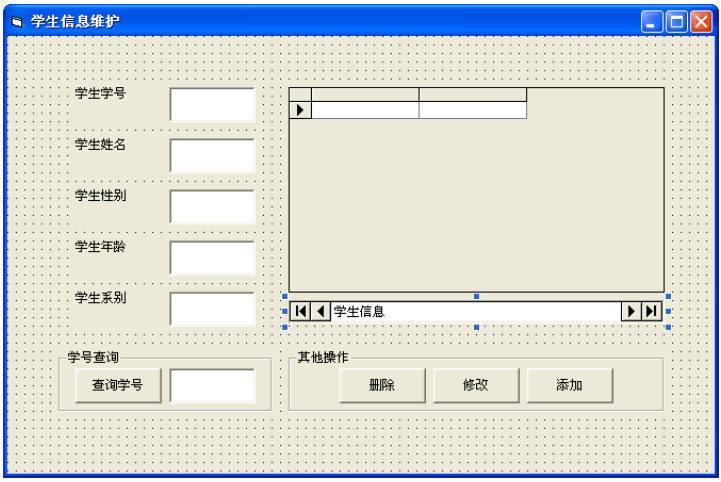


图 11-16　学生信息维护界面

表 11-2　控件属性设置修改

控 件 名 称	属　　性	修　改　值	功　　能
Form2	Caption	学生信息维护	学生信息维护界面
Label1-Label5	Caption	学生学号，学生姓名 学生性别，学生年龄 学生系别	

续表

控 件 名 称	属　性	修 改 值	功　能
Text1-Text5	Text	空	显示相关数据库数据
Text6	Text	空	输入查询学生学号
Frame1	Caption	学号查询	
Frame2	Caption	其他操作	
Command1	Caption	查询学号	查询学号
Command2	Caption	删除	删除相关记录
Command3	Caption	修改	修改相关记录
Command4	Caption	添加	添加相关记录
DataGrid1			列表显示相关数据
Adodc1	Caption	学生信息	

数据控件的添加需要从【工具箱】的【部件】中进行选择，选择添加的这两个控件如图 11-17 所示。添加 Microsoft ADO Data Control 和 Microsoft DataGrid Control 控件。

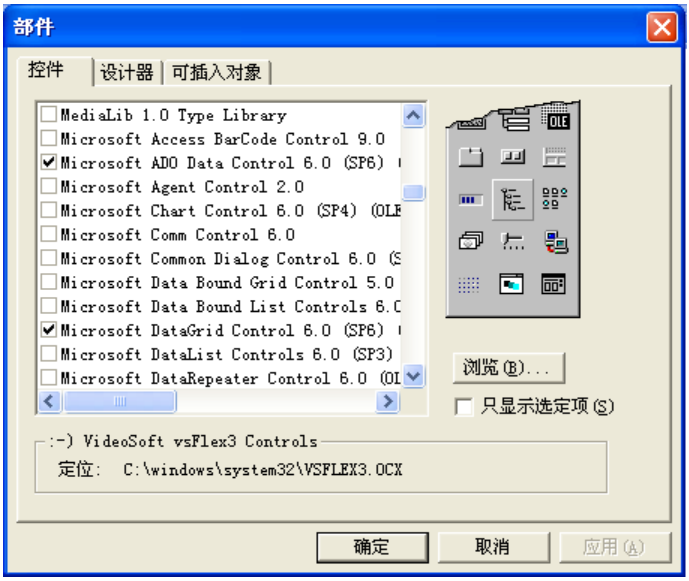


图 11-17　部件添加

11.3.2　Data 数据控件设置和数据绑定

上面已经对【学生信息维护】界面添加了相关的控件，进行了布局，修改了控件的相关属性，下面需要对 Data 数据控件进行相关的数据库连接设置，同时需要对一些控件进行数据的绑定。

（1）Adodc1 控件的设置。选择【Adodc1】控件，单击鼠标右键，选择【ADODC 属性】命令，打开【属性页】对话框，如图 11-18 所示。单击【使用连接字符串】单选按钮，选择【生成】按钮，打开【数据链接属性】对话框，如图 11-19 所示。指定【使用数据源名称】为【myvbsql】，【用户名称】为【sa】，【密码】为空，【输入要使用的初始目录】为【db_stu】，单击【测试连接】

按钮，如果连接成功则单击【确定】按钮，系统会生成连接字符串【Provider=MSDASQL.1；Persist Security Info=False；User ID=sa；Data Source=myvbsql；Initial Catalog=db_stu】。再选择【属性页】的【记录源】选项卡，如图 11-20 所示，设置【命令类型】和【命令文本】。

图 11-18　ADODC 属性页

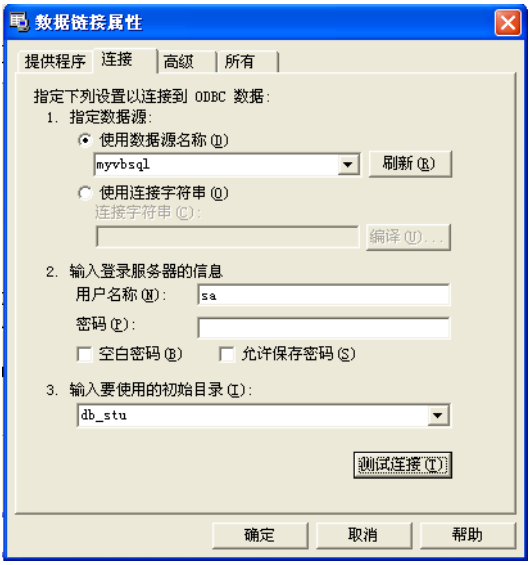


图 11-19　数据连接属性

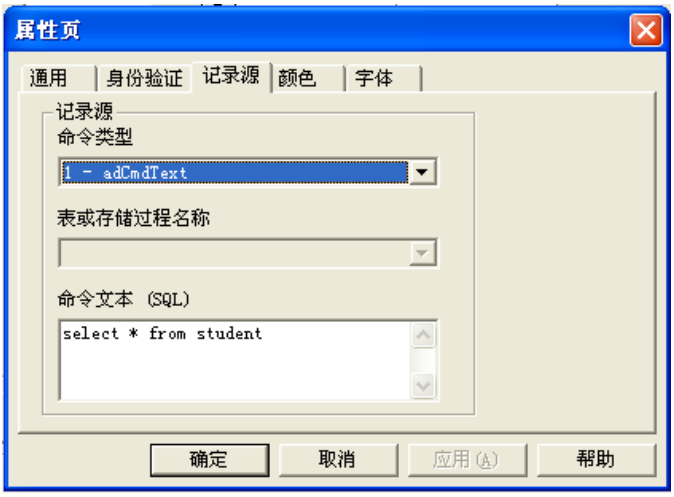


图 11-20　【属性页】的【记录源】选项卡

（2）设置相关控件的数据绑定如表 11-3 所示。

表 11-3　控件数据绑定

控 件 名 称	属　　性	绑　　定	功　　能
DataGrid1	DataSource	Adodc1	列表显示
Text1	DataSource DataField	Adodc1 Sno	绑定到 Sno 字段

续表

控 件 名 称	属　　性	绑　　定	功　　能
Text2	DataSource DataField	Adodc1 Sname	绑定到 Sname 字段
Text3	DataSource DataField	Adodc1 Ssex	绑定到 Ssex 字段
Text4	DataSource DataField	Adodc1 Sage	绑定到 Sage 字段
Text5	DataSource DataField	Adodc1 Sdept	绑定到 Sdept 字段

11.3.3　VB/SQL 数据库代码实现

前面两小节分别完成了控件布局、属性修改、控件数据连接，以及数据绑定的设置，下面就来完成相关控件事件的代码、对窗体的操作，以及对数据库数据的查询、删除、修改和插入等操作。

（1）在 Form1 窗体上，双击【学生信息维护】菜单栏按钮，添加单击【学生信息维护】按钮的事件代码。双击【退出系统】菜单栏按钮，添加单击【退出系统】按钮事件代码。

```
Private Sub S_menu_Click()
Form2.Show
End Sub

Private Sub menu2_Click()
End
End Sub
```

（2）在 Form2 代码窗口添加代码。为 Form2 添加装载和卸载事件代码，以实现数据连接和数据连接的关闭；添加 MakeSqlStr 函数用于创建查询的字符串。

```
Private sqlcon As New ADODB.Connection 'ADO 连接对象
Private sqlres As New ADODB.Recordset 'ADO 数据集对象
Private sqlcmd As New ADODB.Command 'ADO 命令对象
Dim sqlstr As String '查询字符串变量

Private Sub Form_Load()
sqlcon.Provider = "SQLOLEDB"
sqlcon.Open "Server=;DataBase=db_stu;UID=sa;PWD=;""打开连接
End Sub

Private Sub Form_Unload(Cancel As Integer)
sqlcon.Close'关闭连接
End Sub

Public Sub MakeSqlStr()'创建查询字符串函数
```

```
sqlstr = ""
If Trim(Text6.Text) <> "" Then
sqlstr = Trim(Text6.Text)
End If
If Trim(Text6.Text) = "" Then
sqlstr = Text1.Text
End If
End Sub
```

（3）在 Form2 窗口双击【查询学号】按钮，为【查询学号】按钮添加相关的事件代码。

```
Private Sub Command1_Click() '查询
MakeSqlStr
Adodc1.RecordSource = "select * from student where Sno=" & sqlstr
Adodc1.Refresh
End Sub
```

（4）在 Form2 窗口双击【删除】按钮，为【删除】按钮添加相关的数据库数据删除事件代码。

```
Private Sub Command2_Click() '删除
ret = MsgBox("是否要删除" + Adodc1.Recordset("Sno") + "号学生的记录", vbYesNo, "提示")
If ret = vbYes Then
sqlcmd.ActiveConnection = sqlcon
sqlcmd.CommandText = "delete from student where Sno=" + Adodc1.Recordset("Sno")
sqlcmd.Execute
Adodc1.RecordSource = "select * from student"
Adodc1.Refresh
End If
End Sub
```

（5）在 Form2 窗口双击【修改】按钮，为【修改】按钮添加相关的数据库数据修改事件代码。

```
Private Sub Command3_Click() '修改
Dim sqlupstr As String
sqlupstr = "select * from student where Sno=" & Trim(Text1.Text)
sqlres.Open sqlupstr, sqlcon, adOpenDynamic, adLockPessimistic
If Not sqlres.EOF Then
sqlres("Sno") = Text1.Text
sqlres("Sname") = Text2.Text
sqlres("Ssex") = Text3.Text
sqlres("Sage") = Text4.Text
sqlres("Sdept") = Text5.Text
```

```
sqlres.Update
End If
sqlres.Close
Adodc1.RecordSource = "select * from student"
Adodc1.Refresh
End Sub
```

（6）在 Form2 窗口双击【添加】按钮，为【添加】按钮添加相关的数据库数据添加事件代码。

```
Private Sub Command4_Click() '添加
Dim sqlupstr As String
sqlupstr = "select * from student where Sno=" & Trim(Text1.Text)
sqlres.Open sqlupstr, sqlcon, adOpenDynamic, adLockPessimistic
If Not sqlres.EOF Then
Else
sqlres.AddNew
sqlres("Sno") = Text1.Text
sqlres("Sname") = Text2.Text
sqlres("Ssex") = Text3.Text
sqlres("Sage") = Text4.Text
sqlres("Sdept") = Text5.Text
sqlres.Update
End If
sqlres.Close
Adodc1.RecordSource = "select * from student"
Adodc1.Refresh
End Sub
```

（7）运行结果如图 11-21 到图 11-23 所示。

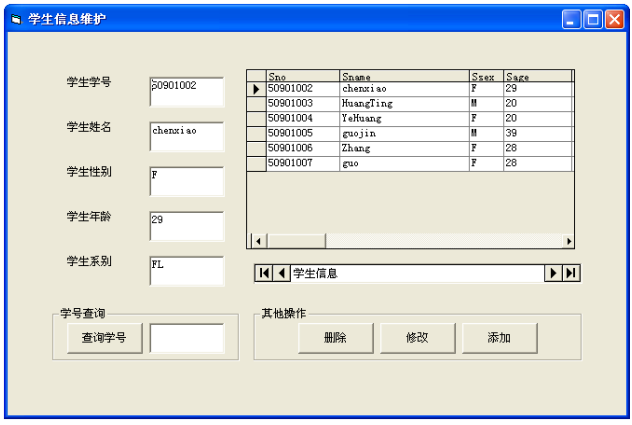


图 11-21　基本运行结果

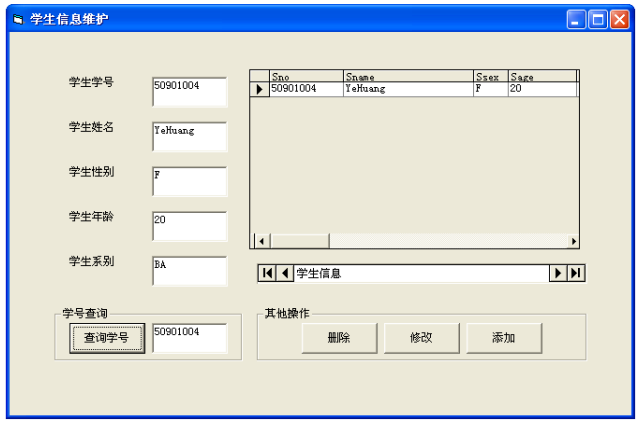


图 11-22　学号查询运行结果

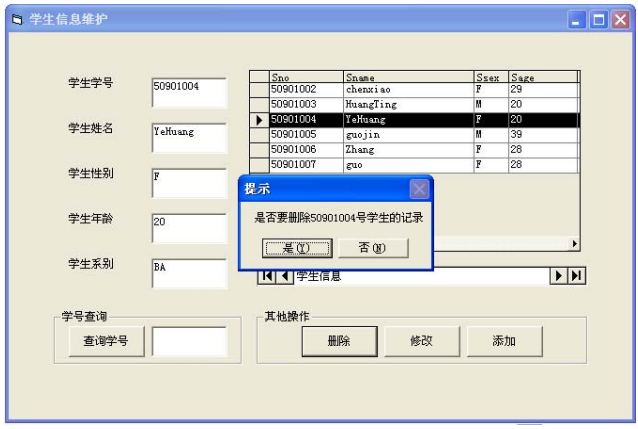


图 11-23　删除操作运行结果

对于修改和添加的功能，读者可以自行进行验证。本节我们实现了对数据库【db_stu】中的数据表【Student】的访问、查询、添加、删除和修改操作，对于其他的两个表，读者可以仿照本节的实例进行代码的实现。课程表和成绩表的界面运行结果如图 11-24 和图 11.25 所示。

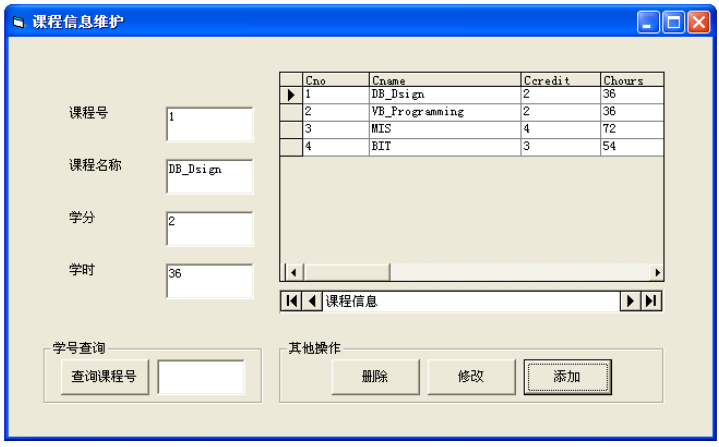


图 11-24　课程信息维护界面运行结果

图 11-25　成绩信息维护界面运行结果

第 12 章　C#.NET/SQL Server 开发与编程

随着.NET 平台的崛起，C#语言作为该平台上的主流语言，在编程中的应用也开始越来越广泛。C#从 C 和 C++演变而来，是一种简单易懂、完全面向对象、类型安全的编程语言，它结合了 Visual Basic 编程的高效率和 C++的强大功能和灵活性，使用更加方便。

本章将简单介绍利用 C#.NET 开发数据库应用程序的方法；介绍 ADO.NET，以及 C#.NET 中对数据库进行访问和操作的关键类；最后以一个学生信息管理应用程序为例，介绍使用 C#.NET 开发 SQL Server 数据库应用程序的思路和方法。

12.1　ADO.NET 概述

关于数据库编程，微软提供了一个统一的数据对象访问模型，在 Visual Studio 6.0 中称为 ADO，在.NET 中则统一称为 ADO.NET，掌握 ADO.NET 就等于掌握了数据库编程的核心。

ADO.NET 是微软.NET 框架中的一种新的数据访问技术，它可以访问很多关系型数据库系统，如 SQL Server 及很多已经具备了 OLE DB 的数据源。以前数据处理主要依赖两层模型，而目前数据处理技术向着多层的结构发展，ADO.NET 正响应了这种需求。

微软在创建 ADO.NET 的时候，具有以下的一些设计目标。

（1）ADO.NET 的设计尽可能和 ADO 保持一致，使得很多 ADO 开发人员不需要从头学习数据库的访问技术。

（2）支持 N 层的编程模式。ADO.NET 为断开 N 层编程环境提供了高级的支持，N 层编程的 ADO.NET 解决方案就是使用了 ADO.NET 里面的 DataSet 组件。

（3）内置的 XML 支持。作为.NET 的一部分，ADO.NET 内置了 XML 支持，使 XML 和数据访问紧密联系了在一起。

ADO.NET 提供了两个核心的组件，DataSet 和.NET 数据提供程序，ADO.NET 组件的主要目的就是要将数据访问从数据操作中分解出来。

12.1.1　ADO.NET DataSet 组件

ADO.NET DataSet 组件为 ADO.NET 提供了断开式结构服务，DataSet 的设计就是为了实现独立于任何数据源的数据访问，所以，我们可以将它用于多种不同的数据源，用于 XML 数据，或用于管理应用程序的本地的数据。DataSet 包含了一个或多个 DataTable 对象的集合，这些对象由数据行和数据列，以及主键、外键、约束和有关 DataTable 对象中数据的关系信息组成。DataSet 是 ADO.NET 结构的主要组件，它是从数据源中检索到的数据在内存中的缓存。DataSet 由一组 DataTable 对象组成，用户可使这些对象与 DataRelation 对象互相关联。DataSet 可将数据和架构作为 XML 文档进行读写。数据和架构可通过 HTTP 传输，并在支持 XML 的任何平台上被任何应用程序使用。

12.1.2　.NET 数据提供程序集

ADO.NET 的另外一个核心组件是.NET 数据提供程序。它可实现数据操作和对数据的快

速访问。.NET 数据提供程序包括了 Connection，Command，DataReader 和 DataAdapter 4 种对象。

- Connection：提供与数据源的连接。
- Command：用户可以访问用于返回数据、修改数据、运行存储过程，以及发送或者检索参数信息的数据库命令。
- DataReader：从数据源中提供高性能的数据流。
- DataAdapter：提供连接 DataSet 对象和数据源的桥梁，并且 DataAdapter 使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使对 DataSet 中数据的更改与数据源保持一致。

12.2　C#.NET 数据库操作关键类

12.2.1　SqlConnection

SqlConnection 对象表示与 SQL Server 数据源的一个唯一的会话。对于客户端/服务器数据库系统，它等效于到服务器的网络连接。SqlConnection 与 SqlDataAdapter 和 SqlCommand 一起使用，可以在连接 Microsoft SQL Server 数据库时提高性能。对于所有第三方 SQL 服务器产品及其他支持 OLE DB 的数据源，请读者使用 OleDbConnection。SqlConnection 对象常用的属性和方法如下。

- Database：获取当前数据库或连接打开后要使用的数据库的名称。
- DataSource：获取要连接的 SQL Server 实例的名称。
- Close：关闭与数据库的连接。这是关闭任何打开连接的首选方法。
- Open：使用 ConnectionString 所指定的属性设置打开数据库连接。

12.2.2　SqlDataAdapter

SqlDataAdapter 是 DataSet 和 SQL Server 之间的桥接器，用于检索和保存数据。SqlData Adapter 通过对数据源使用适当的 Transact-SQL 语句映射 Fill（它可更改 DataSet 中的数据以匹配数据源中的数据）和 Update（它可更改数据源中的数据以匹配 DataSet 中的数据）来提供这一桥接。SqlDataAdapter 常用的属性和方法如下。

- DeleteCommand：获取或设置一个 Transact-SQL 语句或存储过程，以从数据集删除记录。
- InsertCommand：获取或设置一个 Transact-SQL 语句或存储过程，以在数据源中插入新记录。
- SelectCommand：获取或设置一个 Transact-SQL 语句或存储过程，用于在数据源中选择记录。
- UpdateCommand：获取或设置一个 Transact-SQL 语句或存储过程，用于更新数据源中的记录。
- Update：为 DataSet 中每个已插入、已更新或已删除的行调用相应的 INSERT、UPDATE 或 DELETE 语句。

12.2.3　SqlCommand

SqlCommand 表示要对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。常用的 SqlCommand 属性和方法如下。

- Connection：获取或设置 SqlCommand 的实例使用的 SqlConnection。
- CommandText：获取或设置要对数据源执行的 Transact-SQL 语句或存储过程。
- ExecuteNonQuery：对连接执行 Transact-SQL 语句，并返回受影响的行数。

12.3.4　SqlDataReader

SqlDataReader 提供一种从 SQL Server 数据库读取行的只进流的方式。常用的 SqlDataReader 属相和方法如下。

- Connection：获取与 SqlDataReader 关联的 SqlConnection。
- Close：关闭 SqlDataReader 对象。
- Read：使 SqlDataReader 前进到下一条记录。

12.3　C#.NET/SQL Server 编程——学生信息管理系统

本节以学生信息管理系统为例，采用 C#.NET 和 SQL Server 相结合的方式，利用 C#.NET 中的 ADO.NET 技术实现对 SQL Server 数据库的访问、查询、修改、删除和搜索的操作。

在 SQL Server 上已经建立好了学生信息数据库，名称为【db_stu】，它包括 3 个数据表，学生表【Student】，课程表【Course】，以及选课表【SC】，3 个数据表的字段添加和第 11 章相同，这里不对创建过程进行详述。

12.3.1　用户界面设计

（1）打开 VS.NET 2005 如图 12-1 所示，选择【文件】菜单，再选择【新建】中的【项目】命令，打开【新建项目】对话框，如图 12-2 所示。【项目类型】选择【Visual C#】语言，【模板】选择【Windows 应用程序】，项目【名称】为【tf-stu】，【解决方案名称】为【天府学院_学生信息管理】。单击【确定】按钮，创建该解决方案。

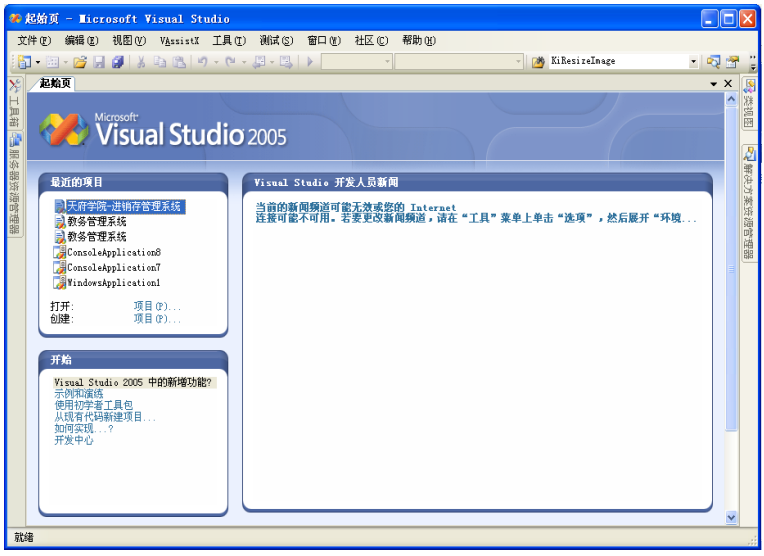


图 12-1　VS 2005 界面

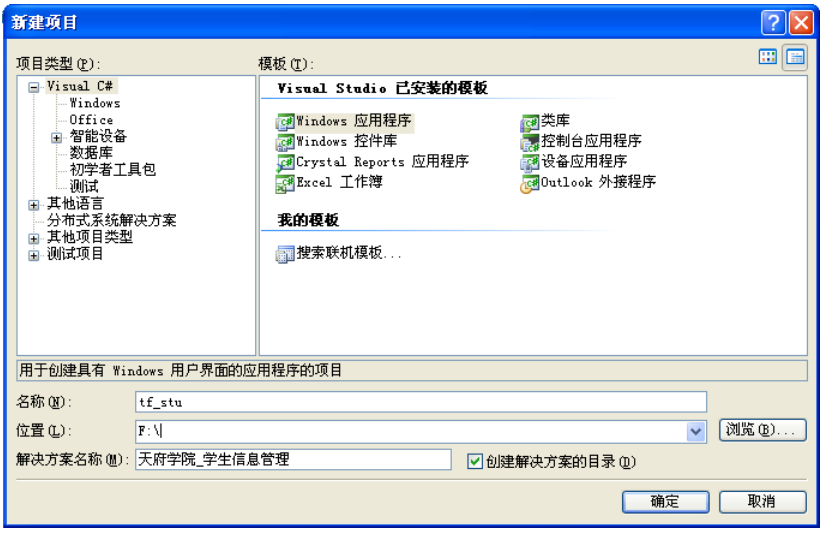


图 12-2　【新建项目】对话框

（2）创建【天府学院_学生信息管理】解决方案的主界面，如图 12-3 所示。各控件的布局及属性修改如表 12-1 所示。

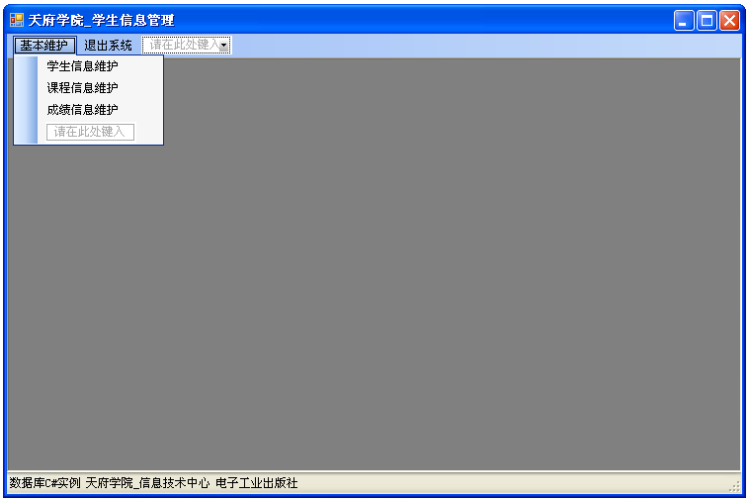


图 12-3　学生信息管理系统主界面

表 12-1　控件属性设置修改

控　件　名	属　　性	修　改　值
Form1	Text	天府学院_学生信息管理
	IsMdiContainer	True
menuStrip1	Text	基本维护（学生信息维护，课程信息维护，成绩信息维护）
	Text	退出系统
statusStrip1	Items	Collection(数据库实例，天府学院_信息技术中心，电子工业出版社)

状态菜单的编辑。选择【statusStrip1】的【Items】属性，单击【Collection】按钮打开【项集合编辑器】对话框，如图 12-4 所示。依次添加 3 个【toolStripStatusLabel】子状态菜单，分别修改其【text】属性为【数据库实例】、【天府学院_学生信息管理】和【电子工业出版社】。

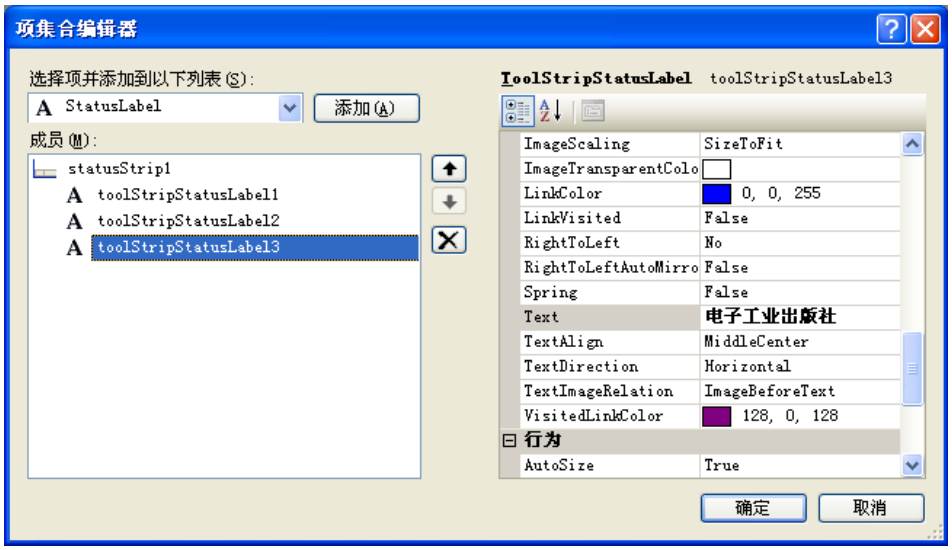


图 12-4　【项集合编辑器】对话框

（3）选择解决方案。单击鼠标右键，选择【添加】按钮，然后选择【新建项】命令，打开【添加新项】对话框，如图 12-5 所示。选择【Windows 窗体】，为应用程序添加一个新的窗体【Form2】。添加相关控件和布局控件如图 12-6 所示。

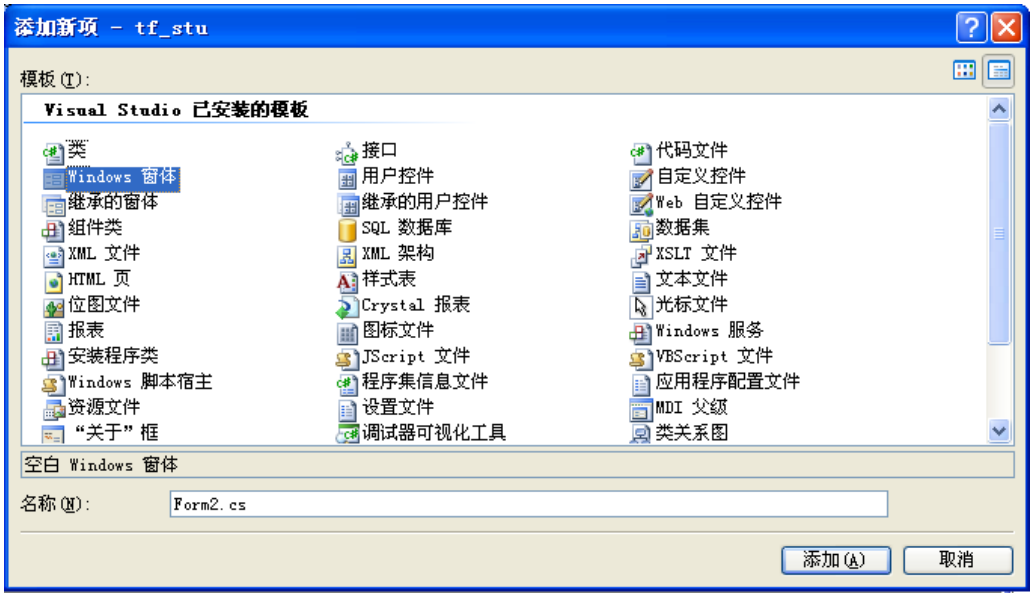


图 12-5　【添加新项】对话框

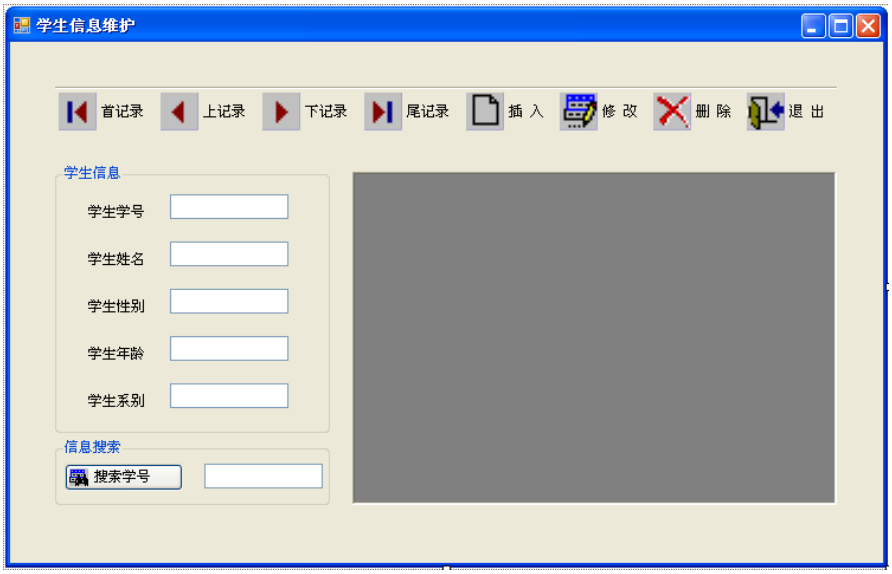


图 12-6　学生信息维护子窗体

控件的属性值修改如表 12-2 所示。需要说明的是，表中需要添加的控件【ToolBar1】和【DataGrid1】需要从工具箱中单击鼠标右键，选择【选择项】，然后选择【ToolBar】和【DataGrid】控件添加，如图 12-7 所示，单击【确定】按钮，即可以添加这两个控件。

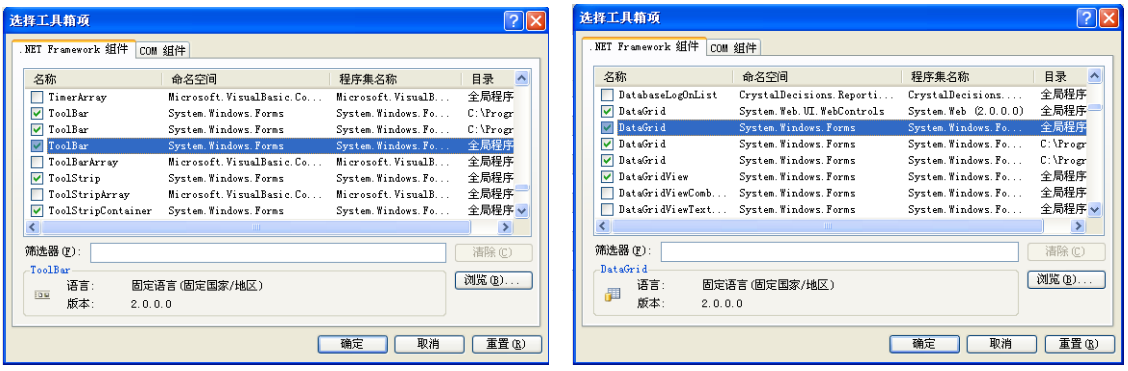


图 12-7　【选择工具箱项】对话框

工具栏的图标按钮的实现是采用【ImageList1】和【ToolBar】这个两个控件共同实现的，首先选择【ImageList1】控件的【Images】属性，单击【Collection】按钮，打开【图像集合编辑器】对话框，按照 Index 从 0～7 的顺序，依次添加 8 个图标，如图 12-8 所示。最后单击【确定】按钮，完成对【ImageList1】控件的设置。然后选择【ToolBar1】控件，选择【Buttons】属性，单击【Collection】按钮，然后再打开【ToolBarButton 集合编辑器】，依次添加成员从 0～7 的 8 个【ToolBarButton】按钮，依次修改其【ImageIndex】属性和【Text】、【ToolTipText】属性，如图 12-9 所示。最后单击【确定】按钮完成对图标工具栏按钮的设置。

表 12-2　控件属性设置修改

控　件　名	属　　性	修　改　值	
Form2	Text	学生信息维护	
ImageList1	Images	Collection（图像集合编辑器）	
ToolBar1	ImageList	imageList1	
	Buttons	ToolbarButtons1	ImageIndex=0,Text=首记录,ToolTipText=首记录
		ToolbarButtons2	ImageIndex=1,Text=上记录,ToolTipText=上记录
		ToolbarButtons3	ImageIndex=2,Text=下记录,ToolTipText=下记录
		ToolbarButtons4	ImageIndex=3,Text=尾记录,ToolTipText=尾记录
		ToolbarButtons5	ImageIndex=4,Text=插　入,ToolTipText=插　入
		ToolbarButtons6	ImageIndex=5,Text=修　改,ToolTipText=修　改
		ToolbarButtons7	ImageIndex=6,Text=删　除,ToolTipText=删　除
		ToolbarButtons8	ImageIndex=7,Text=退　出,ToolTipText=退　出
GroupBox1	Text	学生信息	
GroupBox2	Text	信息搜索	
Label1	Text	学生学号	
Label2	Text	学生姓名	
Label3	Text	学生性别	
Label4	Text	学生年龄	
Label5	Text	学生系别	
Button1	Image	tf_stu.Properties.Resources.搜索（选择资源）	
	ImageAlign	TopLeft	
	Text	搜索	
DataGrid1			
添加：textBox1～textBox6			

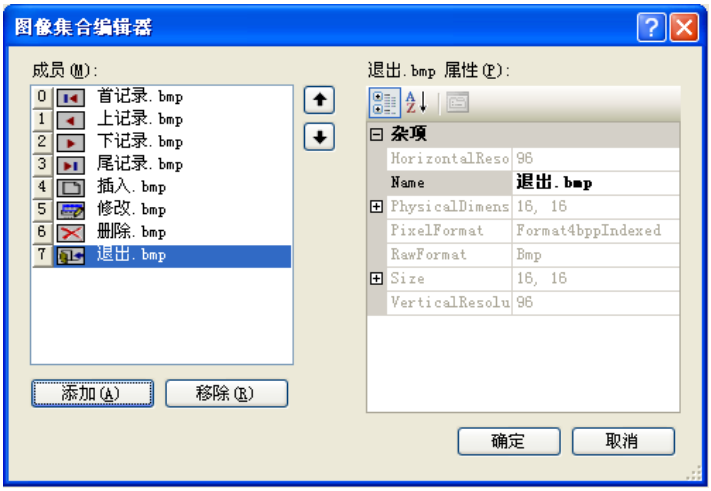


图 12-8　【图像集合编辑器】对话框

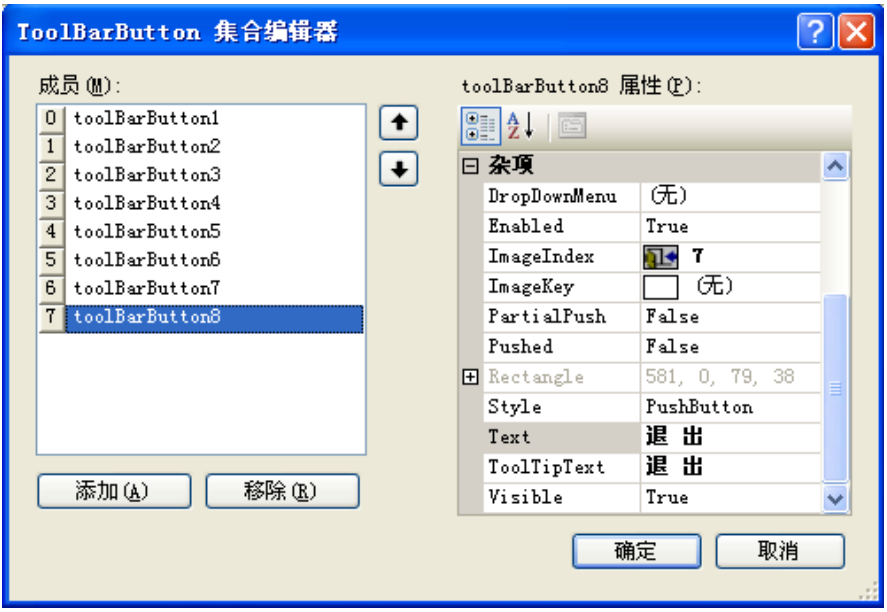


图 12-9　【ToolBarButton 集合编辑器】对话框

12.3.2　C#.NET/SQL 数据库代码实现

（1）双击 Form1 菜单的【学生信息维护】按钮，添加按键事件。

```
//----------------------查询 MDI 子窗体是否存在-------------------------
        private bool checkChildFrmExist(string childFrmName)
        {
            foreach (Form childFrm in this.MdiChildren)
            {
                //用子窗体的 Name 进行判断，如果存在则将它激活
                if (childFrm.Name == childFrmName)
                {
                    if (childFrm.WindowState == FormWindowState.Minimized)
                        childFrm.WindowState = FormWindowState.Normal;
                    childFrm.Activate();
                    return true;
                }
            }
            return false;
        }
//----------------------添加学生信息维护按钮事件-------------------------
    private void  学生信息维护 ToolStripMenuItem_Click(object sender, EventArgs e)
    {
        //通过窗体名称查询该窗体是否已经存在，如存在则显示，否则就新创建一个
```

```
            if (this.checkChildFrmExist("Form2") == true)
            {
                return;
            }
            Form2 newFrm = new Form2();
            newFrm.MdiParent = this;
            newFrm.Show();
        }
    //----------------------添加课程信息维护按钮事件------------------------
        private void  课程信息维护 ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //通过窗体名称查询该窗体是否已经存在，如存在则显示，否则就新创建一个
            if (this.checkChildFrmExist("Form3") == true)
            {
                return;
            }
            Form3 newFrm = new Form3();
            newFrm.MdiParent = this;
            newFrm.Show();
        }
    //----------------------添加成绩信息维护按钮事件------------------------
        private void  成绩信息维护 ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            //通过窗体名称查询该窗体是否已经存在，如存在则显示，否则就新创建一个
            if (this.checkChildFrmExist("Form4") == true)
            {
                return;
            }
            Form4 newFrm = new Form4();
            newFrm.MdiParent = this;
            newFrm.Show();
        }
    //----------------------添加退出系统按钮事件------------------------
        private void  退出系统 ToolStripMenuItem_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

（2）为 Form2 添加命名空间。

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Data.SqlClient;
```

（3）声明私有的类对象。

```
private DataSet myDataSet = new DataSet();//初始化一个新的 DataSet
private CurrencyManager myBind;//用于数据导航控制
```

（4）编写 From2 的 Load 事件代码，连接数据库。

```
//---------编写 From2 的 Load 事件代码，连接数据库----------
        private void Form2_Load(object sender, EventArgs e)
        {
            //与 SQL Server 的连接字符串设置数据库名 db_stu
            string connectionString = "workstation id=localhost;Integrated
                                                Security=SSPI;database=db_stu";
                //与数据库的连接
                SqlConnection myConnection = new SqlConnection(connectionString);
                myConnection.Open();
                string strCom = " SELECT * FROM Student ";
                SqlDataAdapter myCommand = new SqlDataAdapter(strCom, myConnection);
                myCommand.Fill(myDataSet, "Student");
                dataGrid1.DataSource = myDataSet.Tables[0];
                myConnection.Close();
                myBind = (CurrencyManager)this.BindingContext[myDataSet, "Student"];
                //设置 dataGrid1 数据导航控制
                this.dataGrid1.Select(myBind.Count - 1);
                myBind.Position = myBind.Count - 1;//
                this.dataGrid1.CurrentRowIndex = myBind.Position; //移动表头指示图标
                //文本框数据源的绑定
                SetBindings();
            }
//---------设置文本框数据源的绑定----------
             private void SetBindings()
            {
                textBox1.DataBindings.Add("Text", myDataSet, "Student.Sno");
                textBox2.DataBindings.Add("Text", myDataSet, "Student.Sname");
                textBox3.DataBindings.Add("Text", myDataSet, "Student.Ssex");
                textBox4.DataBindings.Add("Text", myDataSet, "Student.Sage");
```

```
            textBox5.DataBindings.Add("Text", myDataSet, "Student.Sdept");
        }
```

（5）设置 ToolBar 按钮事件。

```
//---------设置 ToolBar 按钮事件----------
private void ToolBar1_ButtonClick(object sender, ToolBarButtonClickEventArgs e)
    {
        if (e.Button.ToolTipText == "首记录")
        if (e.Button.ToolTipText == "上记录")
        if (e.Button.ToolTipText == "下记录")
        if (e.Button.ToolTipText == "尾记录")
        if (e.Button.ToolTipText == "插  入")
        if (e.Button.ToolTipText == "修  改")
        if (e.Button.ToolTipText == "删  除")
        if (e.Button.ToolTipText == "退  出")
    }
```

① 设置【首记录】按钮事件。

```
if (e.Button.ToolTipText == "首记录")
{
    this.dataGrid1.UnSelect(myBind.Position); //取消原选中的行
    myBind.Position = 0;
    this.dataGrid1.Select(myBind.Position); //选中当前行
    //移动表头指示图标
    this.dataGrid1.CurrentRowIndex = myBind.Position;
    return;
}
```

② 设置【上记录】按钮事件。

```
if (e.Button.ToolTipText == "上记录")
{
  if (myBind.Position == 0)
      MessageBox.Show("已经到了第一条记录！", "信息提示！",
      MessageBoxButtons.OK, MessageBoxIcon.Information);
  else
  {
      this.dataGrid1.UnSelect(myBind.Position);
      myBind.Position--;
      this.dataGrid1.Select(myBind.Position);
      this.dataGrid1.CurrentRowIndex = myBind.Position;
  }
```

```
        return;
    }
```

③ 设置【下记录】按钮事件。

```
    if (e.Button.ToolTipText == "下记录")
    {
        if (myBind.Position == myBind.Count - 1)
            MessageBox.Show("已经到了最后一条记录！", "信息提示！",
            MessageBoxButtons.OK, MessageBoxIcon.Information);
        else
        {
            this.dataGrid1.UnSelect(myBind.Position);
            myBind.Position++;
            this.dataGrid1.Select(myBind.Position);
            this.dataGrid1.CurrentRowIndex = myBind.Position;
        }
        return;
    }
```

④ 设置【尾记录】按钮事件。

```
    if (e.Button.ToolTipText == "尾记录")
    {
        this.dataGrid1.UnSelect(myBind.Position);
        myBind.Position = myBind.Count - 1;
        this.dataGrid1.Select(myBind.Position);
        this.dataGrid1.CurrentRowIndex = myBind.Position;
        return;
    }
```

⑤ 设置【插入】按钮事件。

```
    if (e.Button.ToolTipText == "插  入")
    {
        try
        {
            myBind.Position = myBind.Count;
            this.dataGrid1.Select(i);
            //判断所有字段是否添完，添完则执行，反之弹出提示
            if (textBox1.Text != "" && textBox2.Text != "" &&
        textBox3.Text != "" && textBox4.Text != "" && textBox5.Text != "")
        {
            //与 SQL Server 的连接字符串设置
```

```
        string connectionString = "workstation
        id=localhost;Integrated Security=SSPI;database=db_stu";
        //与数据库的连接
        SqlConnection myConnection;
        myConnection = new SqlConnection(connectionString);
        myConnection.Open();
        string strInsert = " INSERT INTO Student ( Sno , Sname ,
                           Ssex , Sage , Sdept ) VALUES ( '";
                           strInsert += textBox1.Text + "','";
                           strInsert += textBox2.Text + "','";
                           strInsert += textBox3.Text + "','";
                           strInsert += textBox4.Text + "','";
                           strInsert += textBox5.Text + "')";
    SqlCommand myCommand = new SqlCommand(strInsert, myConnection);
            myCommand.ExecuteNonQuery();
            myConnection.Close();
    myDataSet.Tables["Student"].Rows[myBind.Position].BeginEdit();
    myDataSet.Tables["Student"].Rows[myBind.Position].EndEdit();
        myDataSet.Tables["Student"].AcceptChanges();
        MessageBox.Show("插入成功");
    }
    else
    {
        MessageBox.Show("必须填满所有字段值！", "错误！");
    }
    }
    catch (Exception ed)
    {
       MessageBox.Show("保存数据记录发生 " + ed.ToString(), "错误！");
    }
```

⑥ 设置【删除】按钮事件。

```
    if (e.Button.ToolTipText == "删 除")
    {
       DialogResult r = MessageBox.Show("是否删除当前记录！", "删除当前
            记录！", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
       int ss = (int)r;
       if (ss == 6) // 单击"确定"按钮
       {
           try
           {
```

```
            //与 SQL Server 的连接字符串设置
            string connectionString = "workstation
            id=localhost;Integrated Security=SSPI;database=db_stu";
                //与数据库的连接
                SqlConnection myConnection;
                myConnection = new SqlConnection(connectionString);
                myConnection.Open();
                string strDele = "DELETE FROM Student WHERE Sno = " +
                                                    textBox1.Text;
                SqlCommand myCommand = new SqlCommand(strDele,
                                                    myConnection);
                //从数据库中删除指定记录
                myCommand.ExecuteNonQuery();
                //从 DataSet 中删除指定记录
            myDataSet.Tables["Student"].Rows[myBind.Position].Delete();
              myDataSet.Tables["Student"].AcceptChanges();
              myConnection.Close();
              MessageBox.Show("删除成功");
            }
            catch (Exception ed)
            {
        MessageBox.Show("删除记录错误信息：  " + ed.ToString(), "错误！");
            }
        }
    }
```

⑦ 设置【退出】按钮事件。

```
    if (e.Button.ToolTipText == "退  出")
    {
      this.Close();
    }
```

⑧ 设置【搜索】按钮事件。

```
    //---------设置搜索按钮事件----------
        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                myDataSet.Clear();//刷新数据集
                string str = textBox6.Text;
                string tempStrSQL = "select * from student where Sno=" + str;
```

```
            //与 SQL Server 的连接字符串设置
            string connectionString = "workstation id=localhost;Integrated
                                                Security=SSPI;database=db_stu";
            //与数据库的连接
            SqlConnection myConnection;
            myConnection = new SqlConnection(connectionString);
            myConnection.Open();
        SqlDataAdapter myCommand = new SqlDataAdapter(tempStrSQL,myConnection);
            myCommand.Fill(myDataSet, "Student");
            dataGrid1.DataSource = myDataSet.Tables[0];
            myConnection.Close();
        myBind = (CurrencyManager)this.BindingContext[myDataSet, "Student"];
             this.dataGrid1.Select(myBind.Count - 1);//设置选中
             myBind.Position = myBind.Count - 1;
        }
        catch
        {
            MessageBox.Show("没有相关学号的信息！ ");
        }
    }
```

（6）运行结果如图 12-10 到图 12-14 所示。

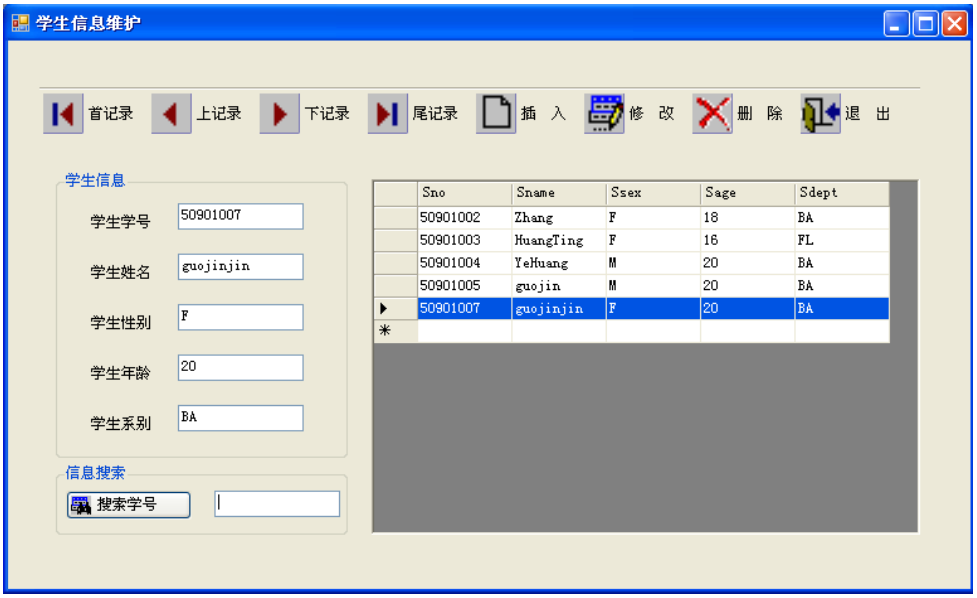


图 12-10　数据库访问运行结果

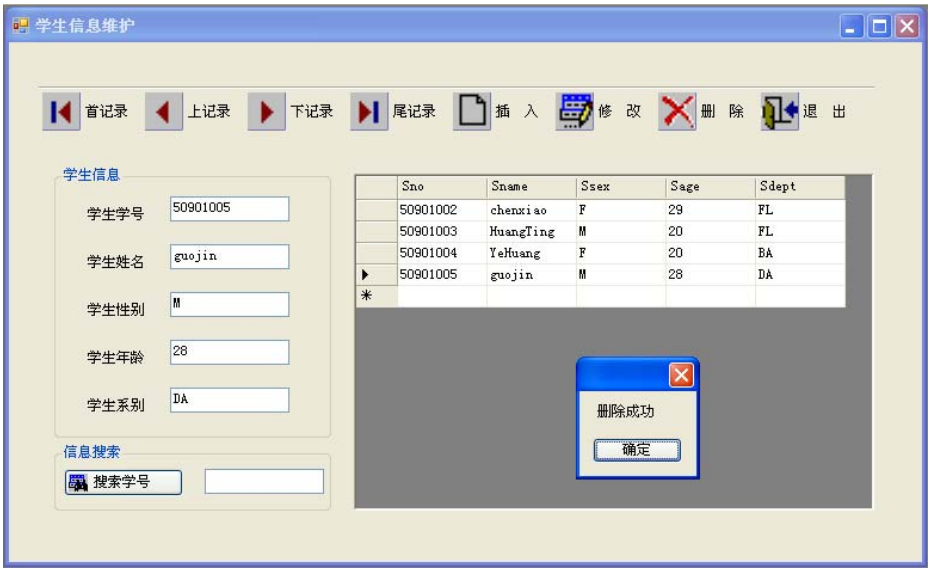


图 12-11　数据记录删除运行结果

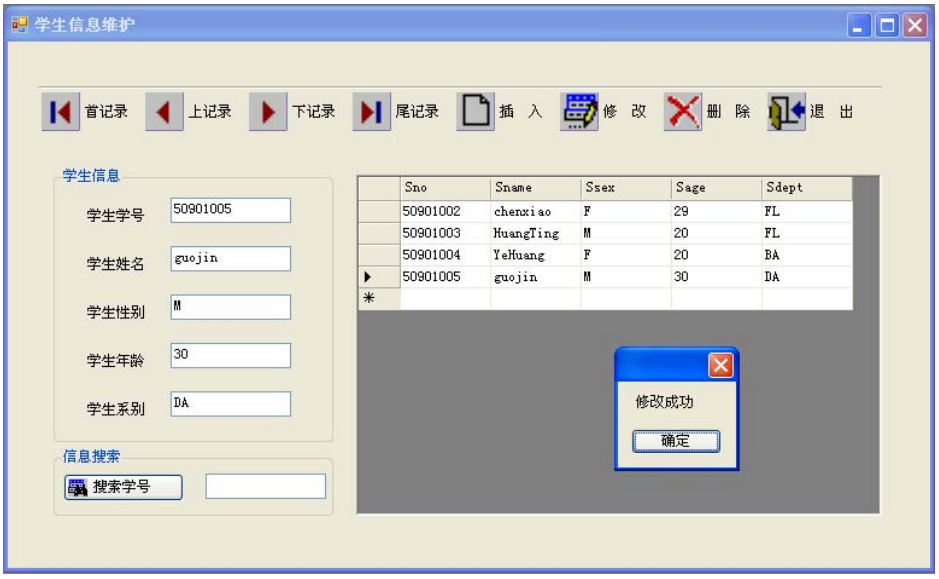


图 12-12　数据记录修改运行结果

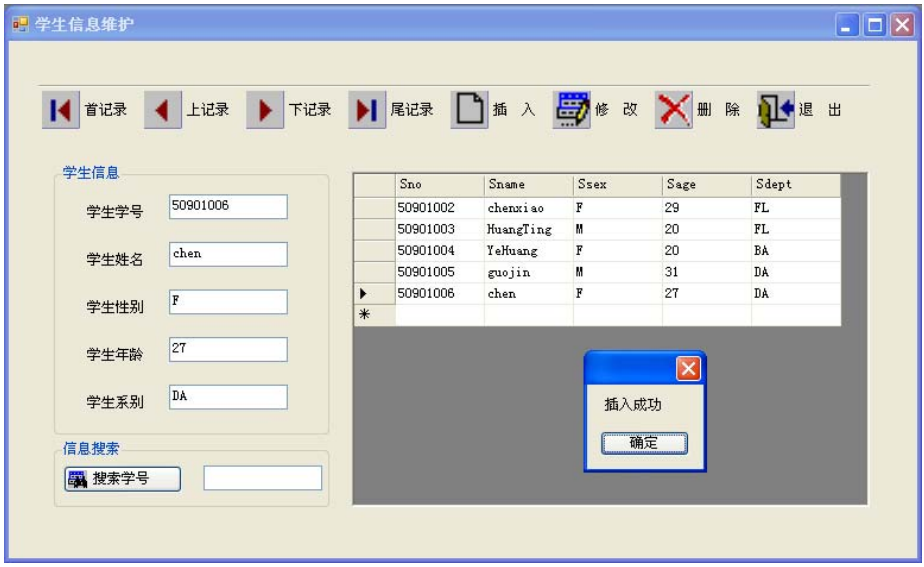


图 12-13　数据插入运行结果

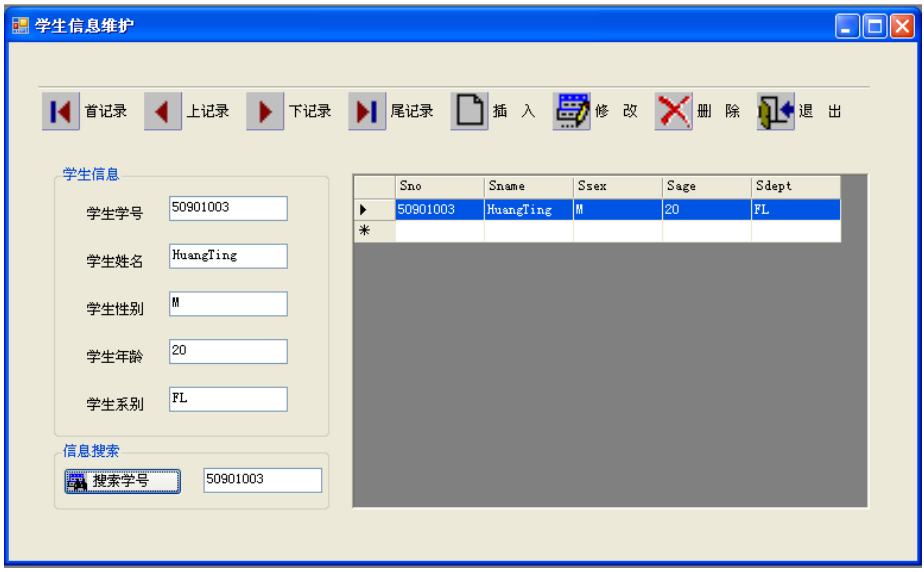


图 12-14　数据运行结果

对于修改和添加的功能，读者可以自行进行验证。本节我们实现了对数据库【db_stu】中的数据表【Student】的访问、查询、添加、删除和修改操作，对于其他的两个表，读者可以仿照本节的实例进行代码的实现。课程表和成绩表的界面运行结果如图 12-15 和图 12-16 所示。

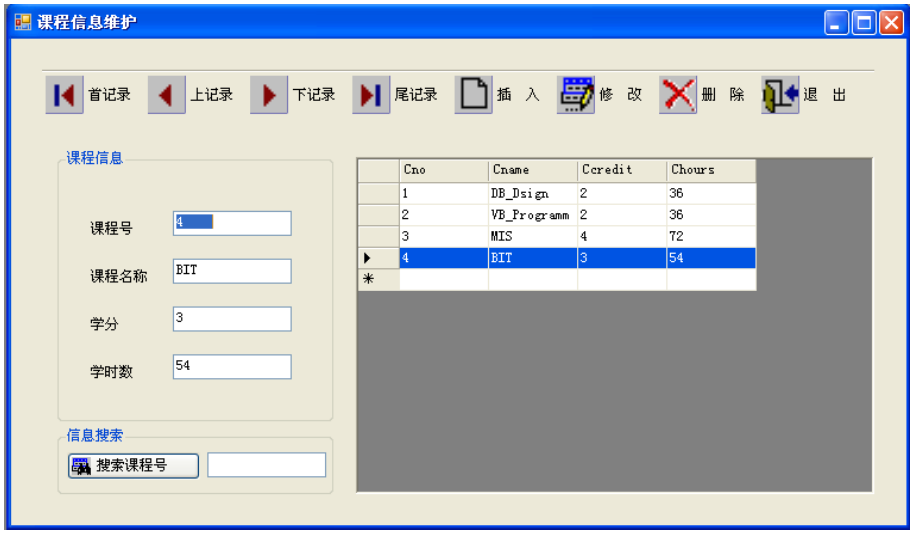


图 12-15　课程信息维护界面运行结果

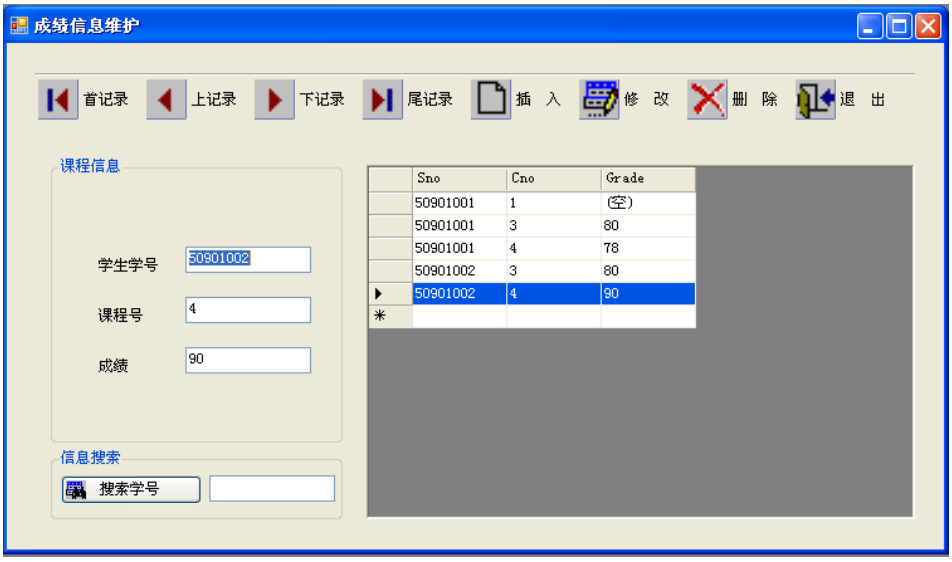


图 12-16　成绩信息维护界面运行结果